MIDDLE EAST TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

CENG 491

INITIAL DESIGN REPORT

SERKAN ÇAĞLAR	**1347285**
BURAK CANSIZOĞLU	**1347244**
SERDAR KOÇBEY	**1250471**
HANİFİ ÖZTÜRK	**1298140**

1 INTRODUCTION

1.1 Purpose of the Document

In the requirement analysis report, we had determined the requirements of our project. With the help of resolving software, hardware, functional and non-functional requirements, we have prepared the initial design report of our project. The initial design report made us comprehend the details and different aspects of the project more clearly and conceptualize the overall product that will be formed accurately. In the design process, we intended to design an effective and modular product that will satisfy the needs and constraints of the project. In the initial design report, we used specific types of UML diagrams like class, sequence and dataflow diagrams. Owing to these diagrams, we have been able to explain the functional, structural and behavioral features of our system.

1.2 Detailed Problem Definition

As the technology evolves, usage of wireless networks has been increased remarkably. In parallel to this, application area of embedded systems integrated with wireless networks has expanded. As a result of this development, wireless sensor networks emerged in the last decade.

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations.[1]

Wireless sensors are far more efficient and feasible than their wired counterparts with respect to their easiness of use, wider range and application areas and less deployment costs. In our project, we aim to establish a wireless sensor network (HSBS_WSN) that will support IEEE

[1]Wikipedia, Wireless Sensor Network, http://en.wikipedia.org/wiki/Wsn, October 2007

802.11 protocol. Supporting IEEE 802.11 protocol of a sensor enables a computer that also supports IEEE 802.11 protocol to receive data from this sensor directly. Most of the wireless sensors in the market support some protocols like ZigBee and IEEE 802.15; however, unfortunately there are too few sensors that support IEEE 802.11 and these types of sensors are too expensive for us to buy. Because of this, we have decided to build our own wireless sensor node (HSBS Sentinel). HSBS Sentinels will have two main advantages. Firstly, they will communicate directly with PCs that support IEEE 802.11. Secondly, they will be more affordable than the other wireless sensors that are available in the market. In the project, there will be a number of HSBS Sentinels which some of them will be in both access point and bridge modes. Moreover, there will be a server that will collect data from HSBS Sentinels and store the data in a database. Furthermore, a user interface that will be used to monitor and process the data will be implemented on this server. HSBS WSNs will be able to be used in various applications like environmental monitoring, habitat monitoring, acoustic detection, seismic detection, military surveillance, inventory tracking, medical monitoring, smart spaces, process monitoring, structural health monitoring, health monitoring and security monitoring.

1.3 Design Constraints

1.3.1 Financial Constraints

In the market, there are a limited number of wireless sensors that support IEEE 802.11 protocol. However, these sensors are not affordable for us. Therefore, we have to design our own wireless sensor nodes.

1.3.2 Manufacturing Constraints

Since we have to build our own wireless node, we are going to obtain some hardware equipments and assemble them. However, we may have some problems while combining these different parts and we cannot foresee the difficulties that we may confront. Furthermore, obtaining these equipments take time. For instance, we ordered SHT15s from USA and we have not received them yet and this is a factor that slows us down.

1.3.3 Ergonomic Constraints

Since hardware interface (PCB) and an AirTies AP-400 will be used in HSBS Sentinels, our wireless nodes will be larger and bulkier than other wireless sensors in the market. This

makes HSBS Sentinels be less ergonomic.

1.3.4 Power Constraints

Because we are using a PCB and an AirTies AP-400 in HSBS Sentinels, we are forced to use the power adapters of the board and AP-400. For this reason, HSBS Sentinels will need a power socket.

1.3.5 Resource Constraints

Because wireless sensor network area is a new and vast area, this subject is in fact currently being researched by universities and institutions world wide. For this reason, we have difficulties in finding enough resources.

1.3.6 Lack of Experience of Team Members

Since the group members have taken a few hardware courses and have little experience on hardware, sometimes it is difficult for us to visualize the details of the project.

1.3.7 Time Constraints

The schedule of the project is determined by the CENG 491 course syllabus. From now on, we have about six months to finalize the project successfully. The detailed design of the project should be finished in a month. Working on the prototype and preparation of it will be handled concurrently and it should be accomplished in one and a half month.

1.4 Design Objectives and Goals

1.4.1 Power

As we stated in the design constraints section, HSBS Sentinels will consist of a PCB and an AirTies AP-400 and both of them have power adapters. In our project, we plan to use only one power supply per an HSBS Sentinel. Only the power adapter of the AP-400 will be used to supply energy to an HSBS Sentinel. The power to the PCB will be provided from AP-400 with the help of a regulator.

1.4.2 Lifetime

The HSBS_WSN should operate properly for a long time. Since, the sensors (SHT15) have CMOSens Technology, they have long-term stability. By the help of this feature of the sensors, the overall product will be able to run for a long time.

1.4.3 Security

Security issue is a big problem for WSNs, however it has been overcome by WPA (802.1x, TKIP, PSK), WPA2 (IEEE802.11i, AES, CCMP), WEP (64/128 bit), MAC filtering and SSID hiding properties of AirTies AP-400.

1.4.4 Accuracy

The HSBS WSN will be able to determine accurate temperature and humidity values by the capability of high-precision measurement of the SHT15s.

1.4.5 Size

We could have been used CENG 336 Embedded Board in HSBS Sentinels. However, then an HSBS Sentinel would be bulky. For this reason, we will implement our own PCB to prevent this situation.

1.4.6 Cost

Existing similar products are expensive to be attained. The wireless node that will be developed by HSBS Smart costs approximately $200 and this price is about one third of the price of the cheapest wireless sensor in the market. Thus, the overall project will be affordable than the existing systems.

1.4.7 Wide Range

Mesh technology enables an AP-400 to function as a repeater. Moreover, an AP-400 can be in both access point and bridge modes. Thanks to these properties of AP-400, HSBS_WSN will operate on a wide range.

2 ARCHITECTURAL DESIGN

2.1 System Hardware Modules

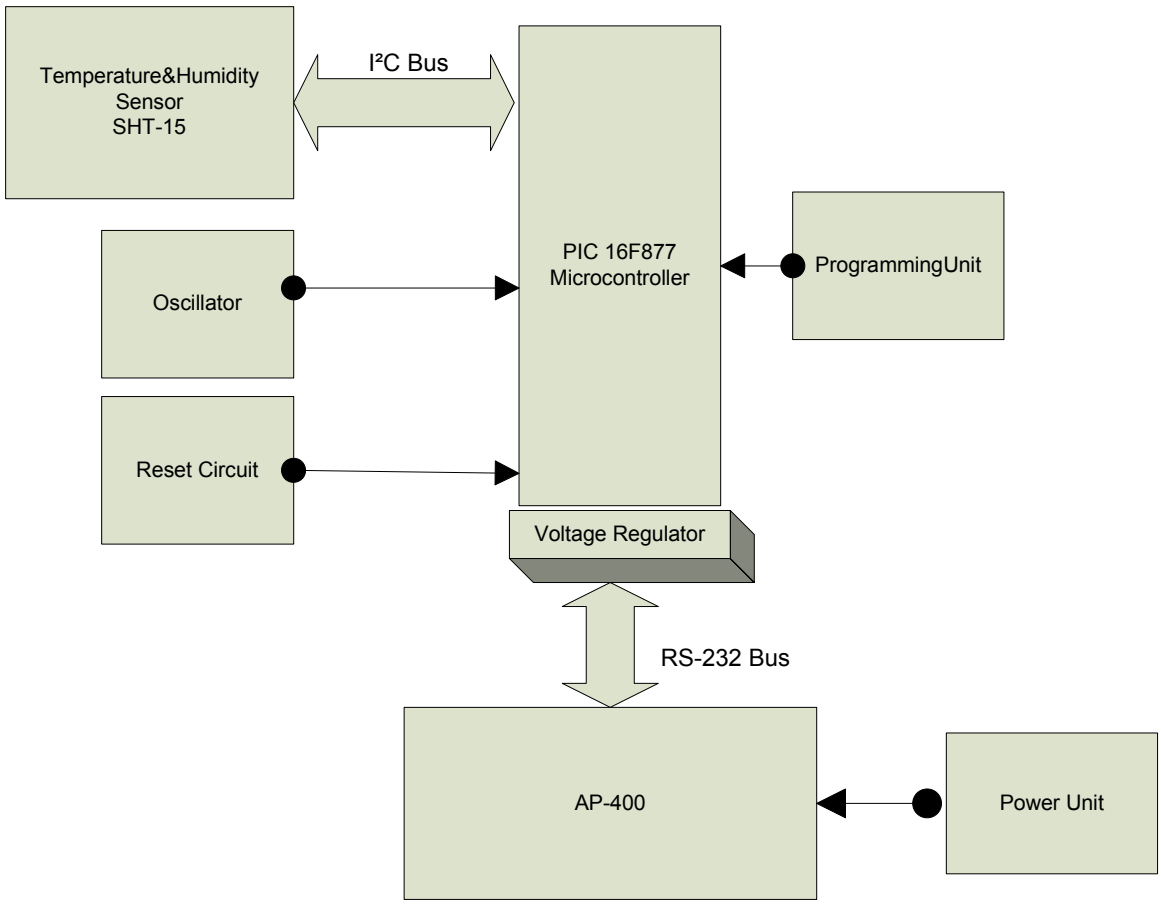


Figure 2-1 Hardware Block Diagram

In Figure 2-1, hardware block diagram is shown. As seen in the figure, the data transfer between SHT15 and PIC 16F877 are provided by I2C Bus. PIC 16F877 is programmed by the Programming Unit whenever needed. Moreover, an oscillator and a reset circuit are essential parts for the proper functioning of PIC 16F877 and are connected to it. The data transfer between PIC 16F877 and AP-400 is provided through RS-232 Serial Bus. Furthermore, a power unit is connected to AP-400. This power unit will supply the power needed by these hardware units. AP-400 operates via 3.3 V and PIC 16F877 operates via 5V. In order to avoid these power differences, a voltage regulator is used between AP-400 and PIC 16F877.

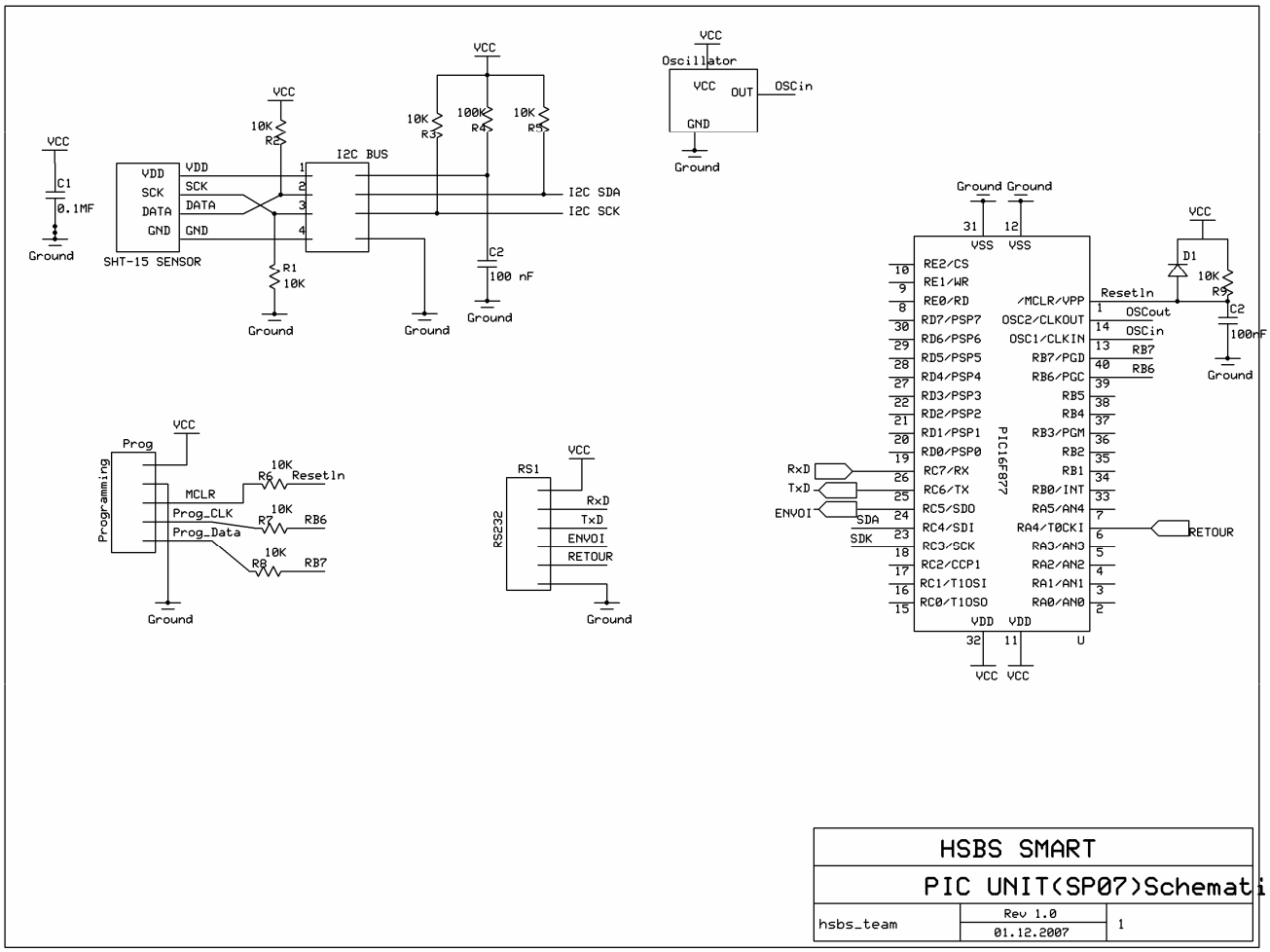


Figure 2-2 General Schematic of SP07

In Figure 2-2, general schematic of SP07 is shown. SP07 is the name for the unit that includes SHT15 and PIC 16F877 for convenience. This schematic shows the connections of the pins in SHT15, Oscillator, Programming Unit, I2C Bus, RS-232 Bus and PIC 16F877. We had a research on the hardware connections in order to be able to draw this schematic; however, due to lack of experience on hardware issues, there may have been some errors in the schematic. However, we are supposed to remove these errors if any, and improve the schematic on the final design report.

2.2 System Software Modules

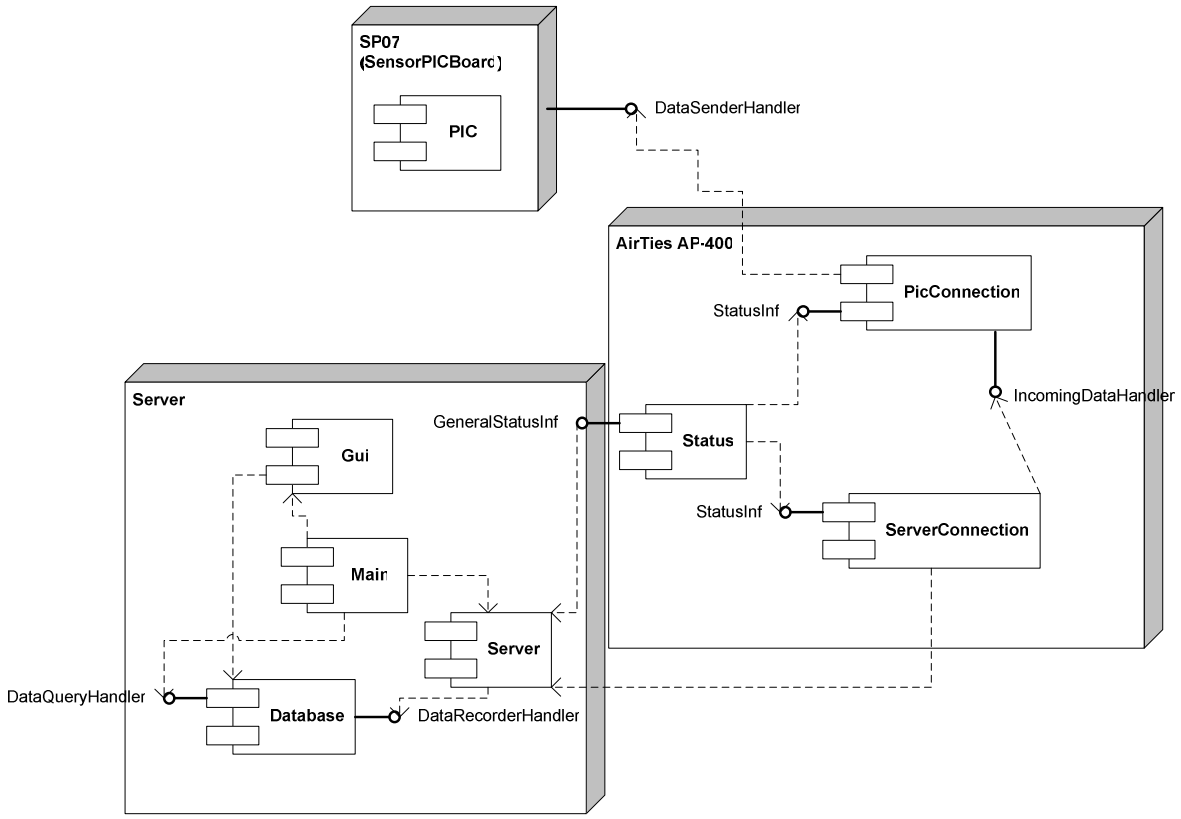


Figure 2-3 Software Component Diagram

The system is composed of three main physical structures; first one is SP07, second AP-400 and third the server. From this point of view, the modules are defined under these three structures hierarchically. The PIC Module which is included in SP07 is responsible for manipulating the sensor read data within the SP07 and provide an interface to AP-400 for data sending. PIC Connection Module is placed in AP-400 and requires an interface from SP07 to retrieve data and process this data internally in AP-400. Status Module which is placed in AP-400 also provides an interface for querying, restarting and turning off AP-400. The Server Connection Module is placed in AP-400 too and operates in order to communicate with Server Module and send the data provided by PIC Connection Module. Server Module provides an interface for Main Module for sending requests to nodes and directs the data to Database Module and node messages to Main Module. GUI Module operates on Server side and is responsible for handling the user interactions like receiving user commands, displaying system outputs, recording user settings. Main Module is placed on server and processes the user commands. It requires an interface from Server Module to send requests and GUI Module to return the outputs. Database Module handles the queries generated by sub

processes from Main Module and Server Module.

2.2.1 PIC Module

PIC Module reads the sensor via I^2C and stores the data into memory banks continuously on each timer interrupt calling `read()` routine. On the other hand if another interrupt is detected on the RS232 port the PIC Module goes into `send()` routine and sends the stored data to the RS232 stream. The routines for stated functions are as follows:

```
read (){
   1. Set the PIC for I²C communication
   2. Set the sensor for reading Humidity
   3. Read the Humidity value into humidity[i] in memory bank
   4. Set the sensor for reading Temperature
   5. Read the Temperature value into temperature[i] in memory bank
   6. Increment i, and take (mod 10)
   7. Return
}

send (){
   1. Set the PIC for RS232 communication
   2. Send ready signal to RS232
   3. Send the humidity[] array contents
   4. Send the temperature[] array contents
   5. Return
}
```

2.2.2 PIC Communication Module

PIC Module sends a read request by RS232 port of the AP-400 which is passed to SP07. It buffers the incoming data from SP07 via RS232 stores it on the memory. Since the shared memory is used among the PIC Communication Module and Server Connection Module; the storing process can be done after PIC Communication Module obtains the lock if it has not already.

```
read_rs232(){
   1. Open com port for RS232 communication
   2. Send read signal to port
   3. Sleep wait for ready signal from port
   4. Buffer the incoming data humidity[], temperature[]
   5. Record the system time
   6. Close the port.
   7. Return.

}

write_shmemory(){
   1. Get the ‘key’ for reading.
   2. Sleep if step 1 fails. Wake up on ‘key’ release go to step 1.
   3. Write buffers humidity[], temperature[] in to shared segment.
   4. Set valid.
   5. Release the ‘key’.
   6. Return.
}
```

2.2.3 Status Module

The Status Module provides an interface for Main Module and requires an interface from Server Module, using functions as `reboot_ap400()`, `turn_off()`, `turn_on()` and `send_info()`. The function `reboot_ap400()` calls the system call reboot after it warns the Server Connection Module. The function `turn_off()` brings the node to a standby state such that no sensor data read and send but node keeps running. The function `turn_on()` takes out the node from standby state. The function `send_info()` sends the current state information to Main Module via TCP/IP protocol. Whenever a turn off or on actions takes place and accomplishes `send_ack()` is called for sending the acknowledge signal to Server Module telling the action was successful. The routines of these functions are as follows;

```
reboot_ap400(){
   1. Send 'rebooting' message to Server Connection Module.
   2. Wait for 5000ms.
   3. Call system call reboot.
}

handle_request(request, requestID){
   1. If request = TURN_OFF then,
   2. Call flag = turn_off();
   3. If request = TURN_ON then,
   4. Call flag = turn_on();
   5. If flag then send_ack(requestID)
   6. Reset flag
   7. If request = REBOOT then,
   8. Call reboot_ap400()
   9. If request = STATUS then,
  10. Call send_info()
  11. Return.
}

turn_off(){
   1. Send 'turning off' signal to Server Connection Module.
   2. Send 'turning off' signal to PIC Connection Module.
   3. Set environment variable state = STANBY
   4. Return True.
}

turn_on(){
   1. Send 'turning on' signal to PIC Connection Module.
   2. Send 'turning on' signal to Server Connection Module.
   3. Set environment variable state = ON
   4. Return True.
}

send_info(){
   1. Open port
   2. send status info via TCP/IP to Server Module
   3. Close port
   4. Return.
}

send_ack(requestID){
   1. Open port
   2. Send 'acknowledge' signal with requestID.
   3. Return.}
```

2.2.4 Server Communication Module

The Server Communication Module requires an interface from Server Module for sending the recorded sensor data on the shared memory. The function `read_shmemory()` reads the recorded data from shared memory written by PIC Communication Module. The function `send_data()` sends the sensor data to Server Module via TCP/IP protocol. The routines of these functions are as follows;

```
read_shmemory(){
   1. Get the 'key' for reading.
   2. Sleep if step 1 fails. Wake up on 'key' release go to step 1.
   3. Read the data from shared segment (humidity[], temperature[]).
   4. Reset valid.
   5. Return.
}

send_data(){
   1. Open port
   2. Send data via TCP/IP to Server Module
   3. Close port.
   4. Return.
}
```

2.2.5 Server Module

The Server Module provides interface for Server Communication Module and Status Module for receiving packages and an interface for `Node` class for node status requests directed to Status Module. Server Module listens two ports, the method `listenStatus()` for receiving status information and acknowledge signals from Status Module and method `listenData()` for receiving sensor data from Server Connection Module. Whenever a package arrives on data port `saveData()` method is called which Database Module provides. Whenever a status package arrives on status port `returnStatus()` method is called which passes the status information to `Node` class according to `nodeID` passed as argument. And whenever an acknowledge signal is detected on status port `returnAck()` method is called passing the message the last action accomplished to `Node` class according to `nodeID` passed as argument. The `sendRequest()` method sends one of the requests `TURN_ON`, `TURN_OFF`, `STATUS`, `REBOOT` to Status Module. The routines of these methods are described as follows;

```
listenStatus(){
   1. Open STATUS PORT for listening.
   2. Wait for package.
   3. Buffer the package.
   4. Parse message_type: status or acknowledge
   5. If message_type is status then,
   6.   Parse nodeID, current_state
   7.   Call returnStatus(nodeID, current_state)
```

```
   8. Else,
   9.   Parse nodeID, request_id
  10.   Call returnAck(nodeID, requestID)
  11. Go to Step 2.
}

listenData(){
   1. Open DATA PORT for listening.
   2. Wait for package.
   3. Buffer the package.
   4. Parse nodeID, dataTemp[], dataHum[], time.
   5. Call DB.saveData(nodeID, dataTemp[], dataHum[], time)
   6. Go to Step 2.
}

returnStatus(nodeID, current_state){
   1. Pass message to Node class with nodeID and current_state
   2. Return.
}

returnAck(nodeID, requestID){
   1. Pass message to Node class with nodeID and requestID
   2. Return.
}

sendRequest(nodeID, request, requestID){
   1. Open port
   2. Retrieve IP of nodeID
   3. Pass request to specified nodeID,
   4. Return.
}
```

2.2.6 Database Module

The Database Module provides interfaces for Main Module and Server Module for querying and updating the database. Methods as `connectDatabase()`, `createTable()`, `insertIntoTable()`, `dropTable()`, `updateTable()` are offered to system. The database module of the system will be taken from an existing API; therefore the methods are not going to be described in detail here. Class diagrams can be seen for further information.

2.2.7 Main Module

The Main Module requires interfaces from Server Module, Database Module and GUI Module for actions like changing node status, fetching data from database, collecting messages and regulating the data for outputting. The base class of the Module is `Node` class. The methods `getStatus()`, `changeNodeState()`, `rebootNode()`, `acknowledge()` are offered as interface. The routines of these methods are as follows;

```
getStatus(){
   1. If status = UNREACHABLE then,
   2.    Return.
   3. Else,
   4.    Generate requestID
   5.    Call server.sendRequest(this.nodeID, STATUS, requestID)
```

```
   6. Return.
}

changeNodeState(){
   3. If Status = ON then,
   4.    Generate requestID
   5.    Call server.sendRequest( this.nodeID, TURN_OFF, requestID)
   6. Else if Status = OFF then,
   7. Else if Status = ON then,
   8.    Generate requestID
   9.    Call server.sendRequest( this.nodeID, TURN_ON, requestID)
  10. Return.
}

rebootNode(){
   1. If Status = UNREACHABLE then,
   2.    Return.
   3. Else,
   4.    Generate requestID
   5.    Call server.sendRequest(this.nodeID, REBOOT, requestID)
   6. Return.
}

acknowledge(requestID){
   1. Pop requestID from activeRequestList[]
   2. Return.
}
```

3 CLASS DIAGRAMS

3.1 Diagram

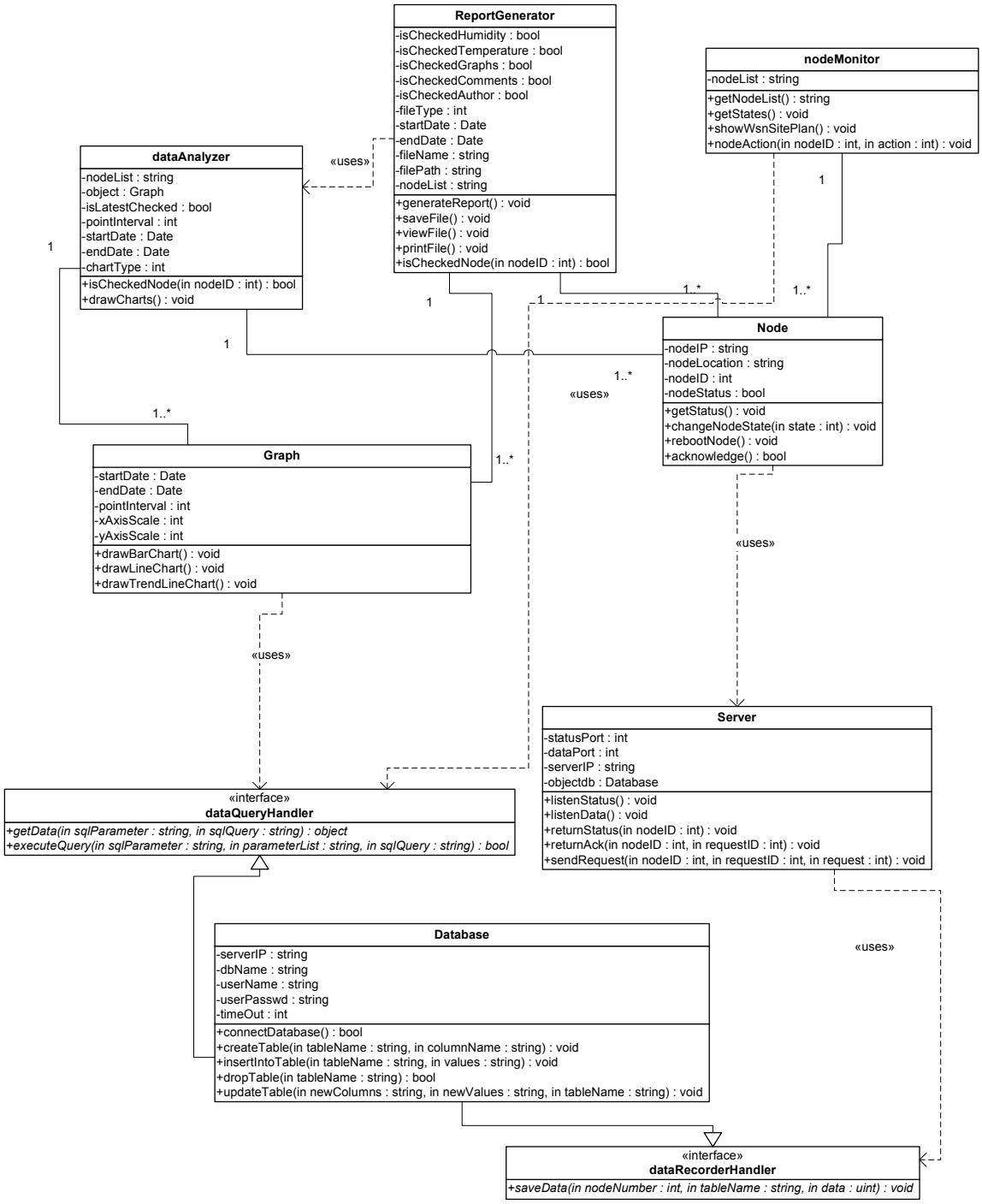


Figure 3-1 Class Diagram

3.2 Class Tables

NodeMonitor				
Attributes	**Attribute Name**	**Type**	**Description**	
	nodeList	Node[ ]	Holds all the information about sensors in the database	
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	getNodeInfo	Node[ ]	Void	Extracts all the information about sensors in the database
	getStateofList	Void	Void	Determines the states of all sensors with the help of server module
	showWsnSitePlan	Void	Void	Displays the existing site plan
	changeStates	Bool	Int	Changes the state of the node

DataAnalyzer			
Attributes	**Attribute Name**	**Type**	**Description**
	nodeList	Node[ ]	Holds all the information about sensors in the database
	Gui	Object GUI	

	isLatestChecked	Bool	Holds the value of the Latest checkbox	
	pointInterval	Int	Sets the x axis interval	
	startDate	Date	Start date of the measured data	
	endDate	Date	End date of the measured data	
	chartType	Int	Holds the chart type	
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	isCheckedNode	Void	Void	Determines whether checkbox of a node is checked or not
	drawCharts	Void	Void	Draws the charts according to user preference

ReportGenerator			
Attributes	**Attribute Name**	**Type**	**Description**
	nodeList	Node[ ]	Holds all the information about sensors in the database
	isCheckedTemperature	Bool	Holds the value of temperature checkbox
	isCheckedHumidity	Bool	Holds the value of humidity checkbox
	isCheckedGraphs	Bool	Holds the value of graph checkbox

	isCheckedComments	Bool	Holds the value of comments checkbox	
	isCheckedAuthor	Bool	Holds the value of author checkbox	
	fileType	Int	Holds the value of file type dropdownlist	
	startDate	Date	Holds the value of start from dropdownlist	
	endDate	Date	Holds the value of end dropdown list	
	fileName	String	Holds the value of file name textbox	
	filePath	String	Holds the value of save to textbox	
	filePreview	String	Holds the value of text version of generated data	
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	generateReport	Void	Void	Shows the data in preview textbox with respect to user preferences
	browsePath	Void	String	Browse the path of the file to be saved
	saveFile	Void	Void	Saves the generated data
	viewFile	Void	Void	Views the print preview of the generated data
	printFile	Void	String	Prints out the generated data

Graph				
Attributes	**Attribute Name**	**Type**	**Description**	
	startDate	Date	Start date of the measured data	
	endDate	Date	End date of the measured data	
	pointInterval	Int	Holds the interval of x axis selected by the user(minutely, hourly, daily)	
	xAxisScale	Int	Holds the interval of x axis scale	
	yAxisScale	Int	Holds the interval of y axis scale	
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	drawBarChart	Void	Void	Draws the bar chart of requested data
	drawLineChart	Void	Void	Draws the line chart of requested data
	drawTrendLine	Void	Void	Draws the line chart of requested data

Node			
Attributes	**Attribute Name**	**Type**	**Description**

	nodeIP	String	Holds IP of the node	
	nodeLocation	String	Holds the location of the node	
	nodeID	Int	Holds the unique id of the node	
	nodeStatus	Bool	Holds the state of the node	
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	getStatus	Void	Void	Sends a request to the server module to get the state of the node
	changeNodeState	Void	Int	Sends a request to the server module to change the state of the node
	rebootNode	Void	Void	Sends a request to the server module to reboot the node
	acknowledge	Bool	Void	Request the acknowledge message of the requested actions

Server			
Attributes	**Attribute Name**	**Type**	**Description**
	statusPort	Int	Holds the port number for the status
	dataPort	Int	Holds the port number for the incoming data

	serverIP	String	Holds the unique id of the server	
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	listenStatus	Void	Void	Listens the status port
	listenData	Void	Int	Listens the data port
	recordData	Void	Void	Records the data which is buffered from the data port
	returnStatus	Bool	Int	Returns the state of the node buffered from the status port to whom it requested it
	returnAck	Void	Int	Returns the acknowledgement of the requests.
	sendRequest	Void	Int, Int, Int	Sends a request to the Status Module in HSBS Sentinel in order to learn the state, turn on, turn off or reboot.

Database			
Attributes	**Attribute Name**	**Type**	**Description**
	serverIP	String	Holds the IP of the database server
	dbName	String	Holds the name of the database
	username	String	Holds the username of the user who will login to the database

	userPasswd	String	Holds the password of the user who will login to the database	
	timeOut	Int	Holds the connection timeout value	
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	connectDatabase	Bool	Void	Creates a connection to the database
	createTable	Void	String, String[]	Creates a table in the database
	insertIntoTable	Void	String, String[],	Inserts a values into a table as a row
	dropTable	Void	String	Deletes the table
	updateTable	Void	String[], String[], String	Updates the table with the new values

dataRecorderHandler <<interface>>				
Methods	**Method Name**	**Return**	**Arguments**	**Description**
	saveData	Void	Int, String, Uint	Saves the data which is coming from the server module

dataQueryHandler <<interface>>

Methods	Method Name	Return	Arguments	Description
	getData	DataTable	String, String	Gets the result of the query from the database
	executeQuery	Bool	String, String, String	Executes the requested query

4 SEQUENCE DIAGRAMS

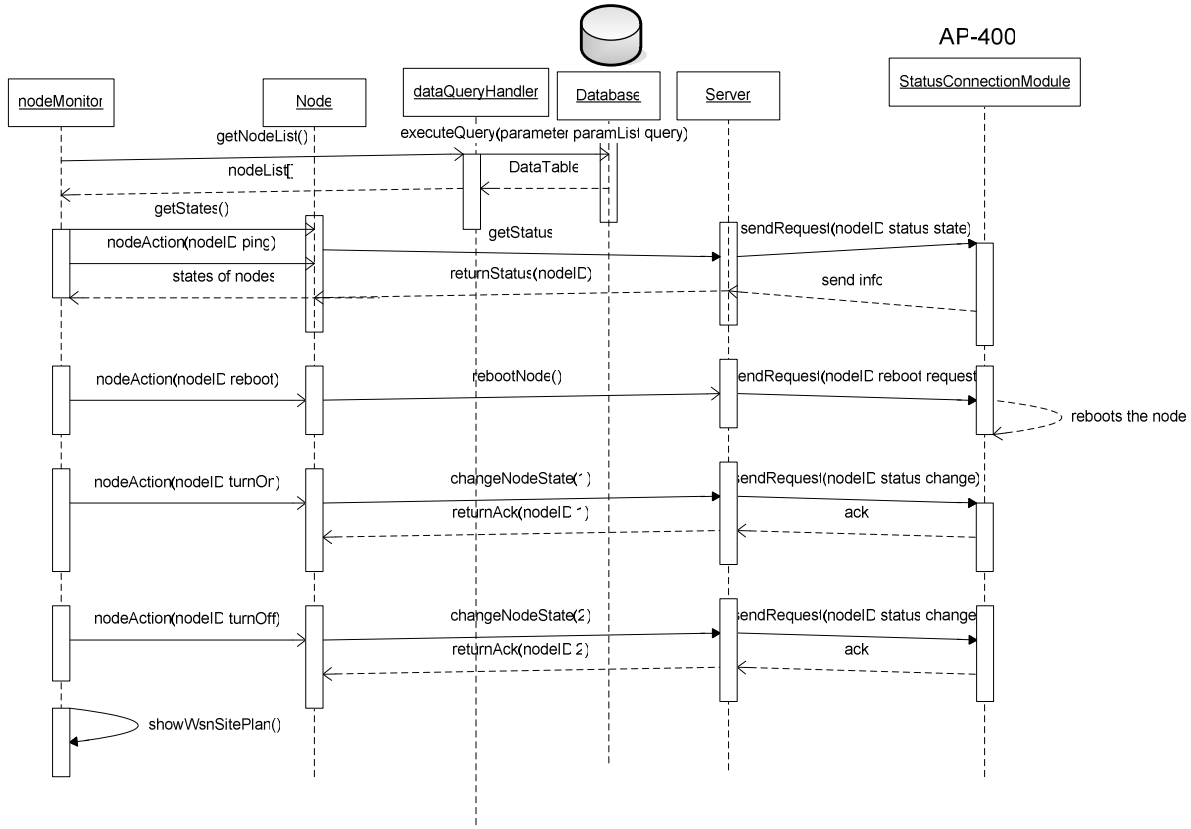


Figure 4-1 Node Monitor

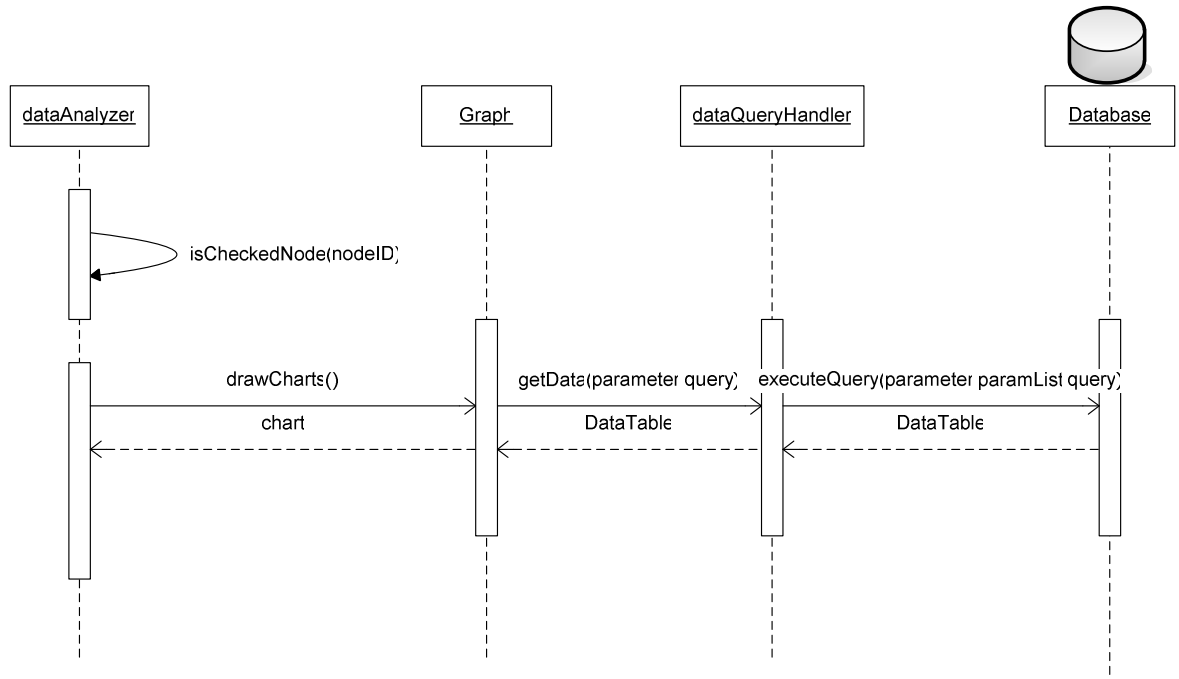


Figure 4-2 Data Analyzer

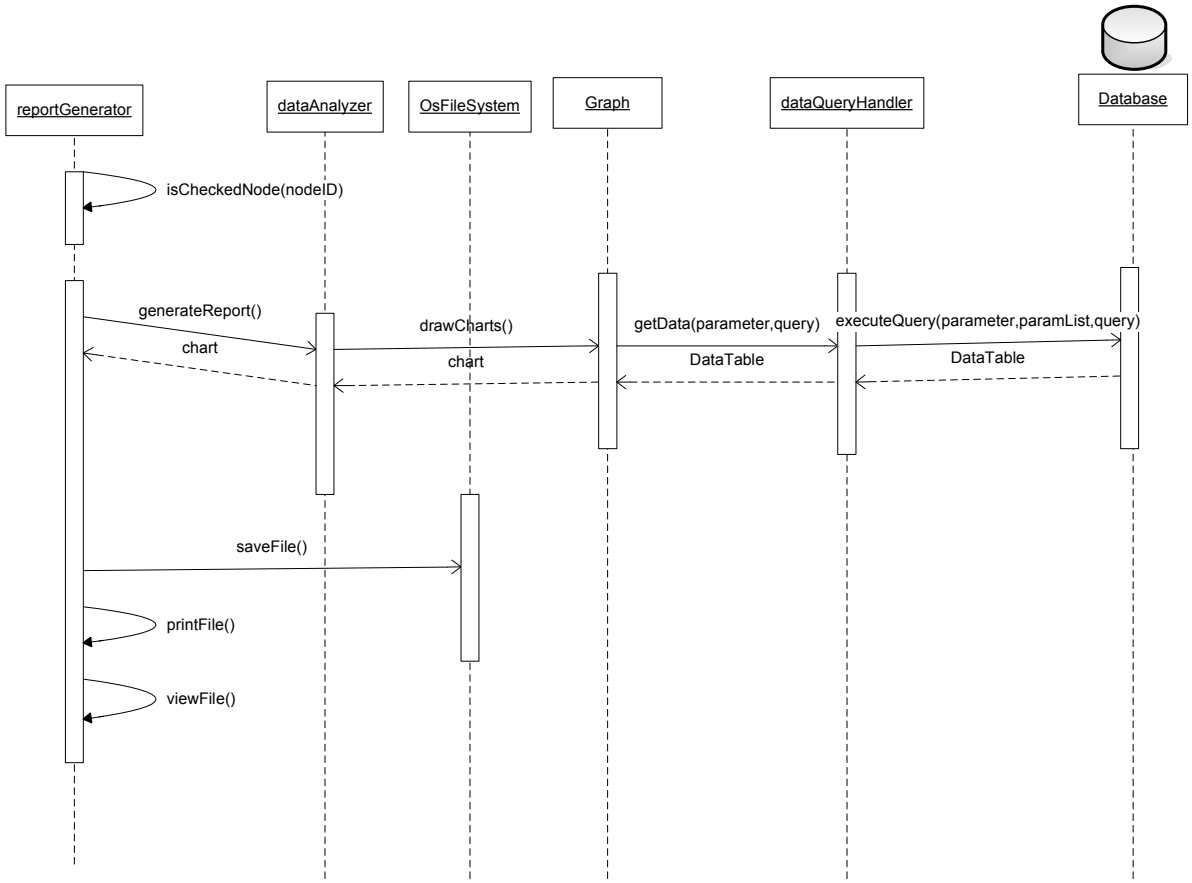


Figure 4-3 Report Generator

5 USER INTERFACE

The HSBS WSN Soft which will run on the server constitutes the user interface of the project. The user interface has three parts namely Node Monitor, Data Analyzer and Report Generator. The following text describes these three features in detail.

5.1 Node Monitor

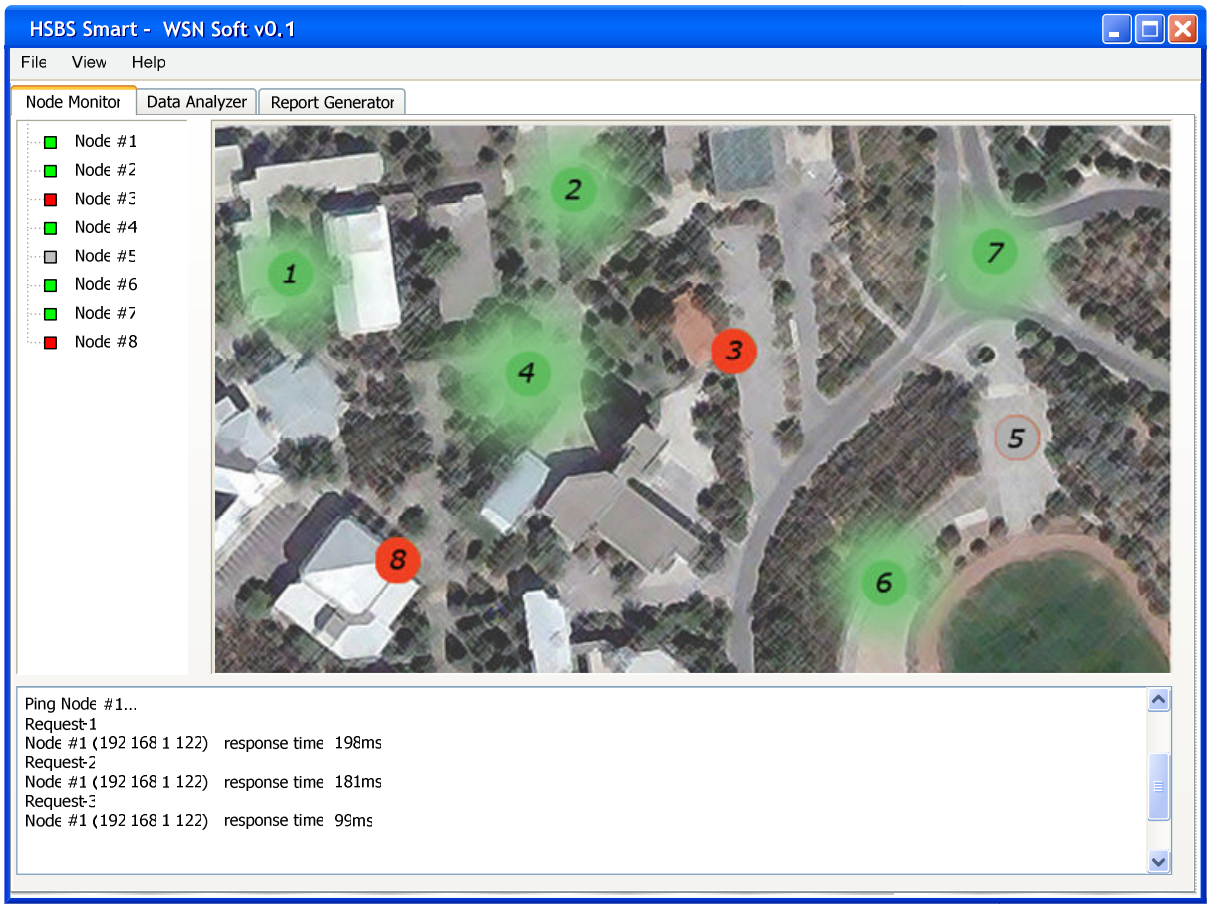


Figure 5-1 Node Monitor

Node Monitor tab has three panels as can bee seen in the Figure-5-1; node list, location map and activation output.

The left side panel lists the nodes with the names user has defined while initiating the system. Little squares before the node names illustrates the status of the node in such a way that "green" square means the node is alive, "red" square means node is turned off, "gray" square means node is unreachable. If a right click mouse event occurs on a node in this panel a menu shows up as shown in the Figure-5-2 allowing user to trigger four actions; ping the node, turn

the node off, turn the node on and reboot the node.

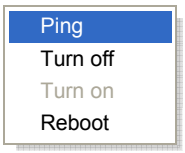


Figure 5-2 Right Click Menu

The location map illustrates the node locations according to their definition to the system. This part is only for only demonstrating the overall view of the system, no interactions are provided to user. The colored circles represent the positions of the nodes. The colors are captured from the left side node list panel.

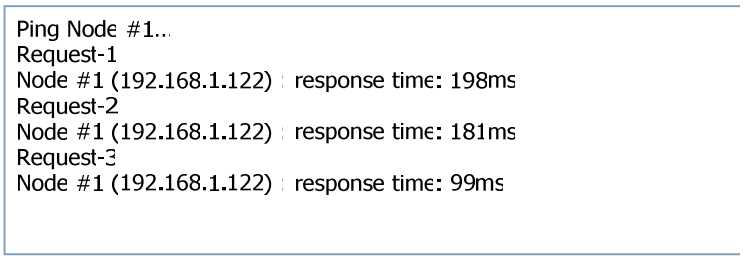


Figure 5-3 Activation Output

The activation output panel in Figure 5-3 displays the messages of system responses the user actions over nodes. The displayed text also recorded into “ActivationLog.txt” in file system.

5.2 Data Analyzer

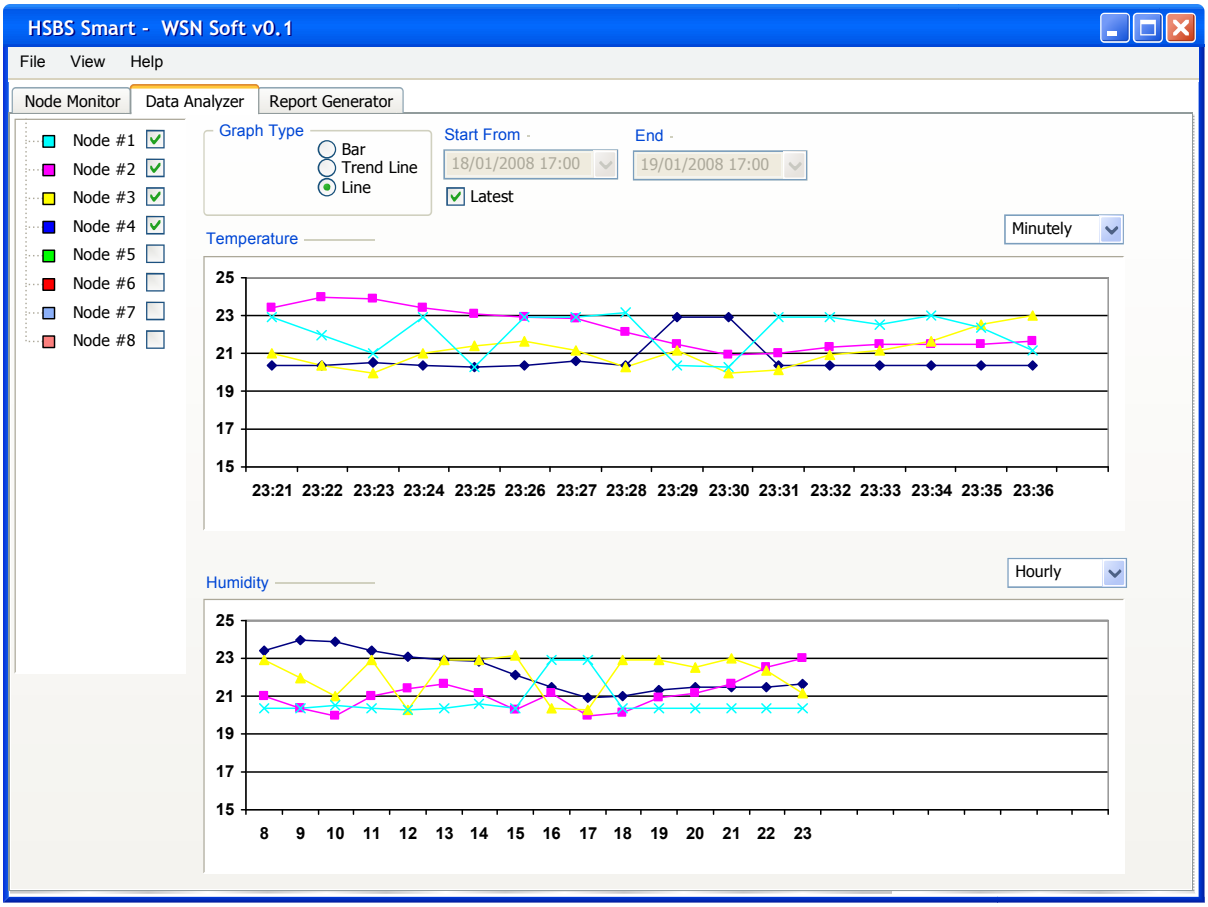


Figure 5-4 Data Analyzer

Data Analyzer tab allows user to see the recorded data on graph illustrations. The nodes defined in the system are listed in the left side panel and specified types of graphs are plotted on the window according to the options specified by user using options panel.

The node list on the left side panel includes checkboxes for each node and user can add or remove the nodes to be displayed in the graphs. Different colors are assigned to each node to make the graphs more understandable.

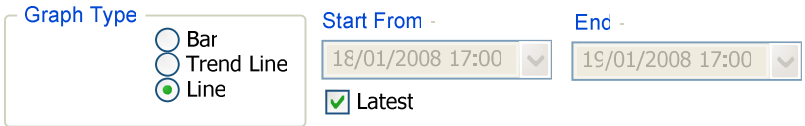


Figure 5-5 Graph Options Panel

The options panel in the Figure 5-5 consists of two parts, first the graph type radio group and second part is time panel to set the interval of the graph to be plotted. In the graph type radio group there are three types of graphs as, bar, trend line and line and if a change event occurs

the graphs are redrawn. The time panel allows user to define the start and end points of the graph in means of time. The user also can select to display latest time interval and let the program select the interval.

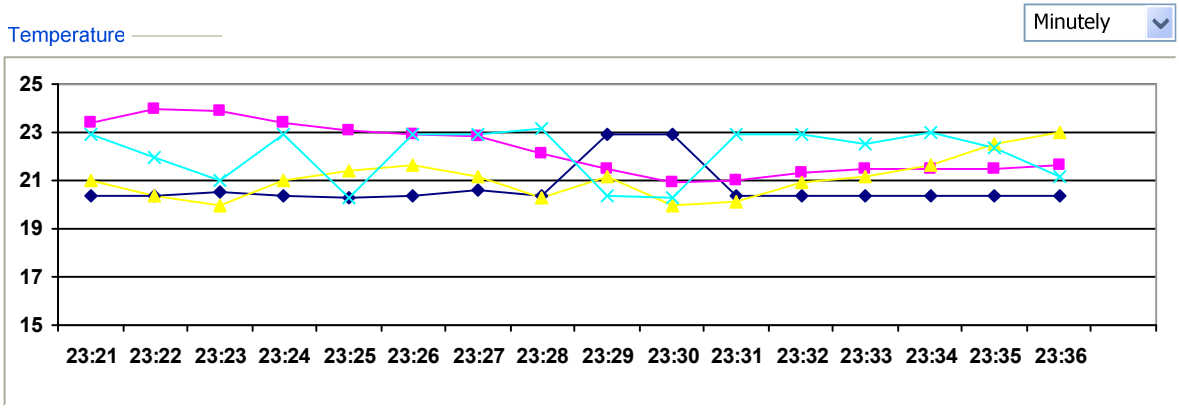


Figure 5-6 Plotted Graph

There always two graphs are drawn a temperature and a humidity like in Figure 5-6. User can set the frequency of the data shown from the combo box on the right top of each graph. The minutely, hourly and daily options are listed here and if user changes this setting the graph related is redrawn. Each line or bar on the graph is related to a node on the left side node list panel.

5.3 Report Generator

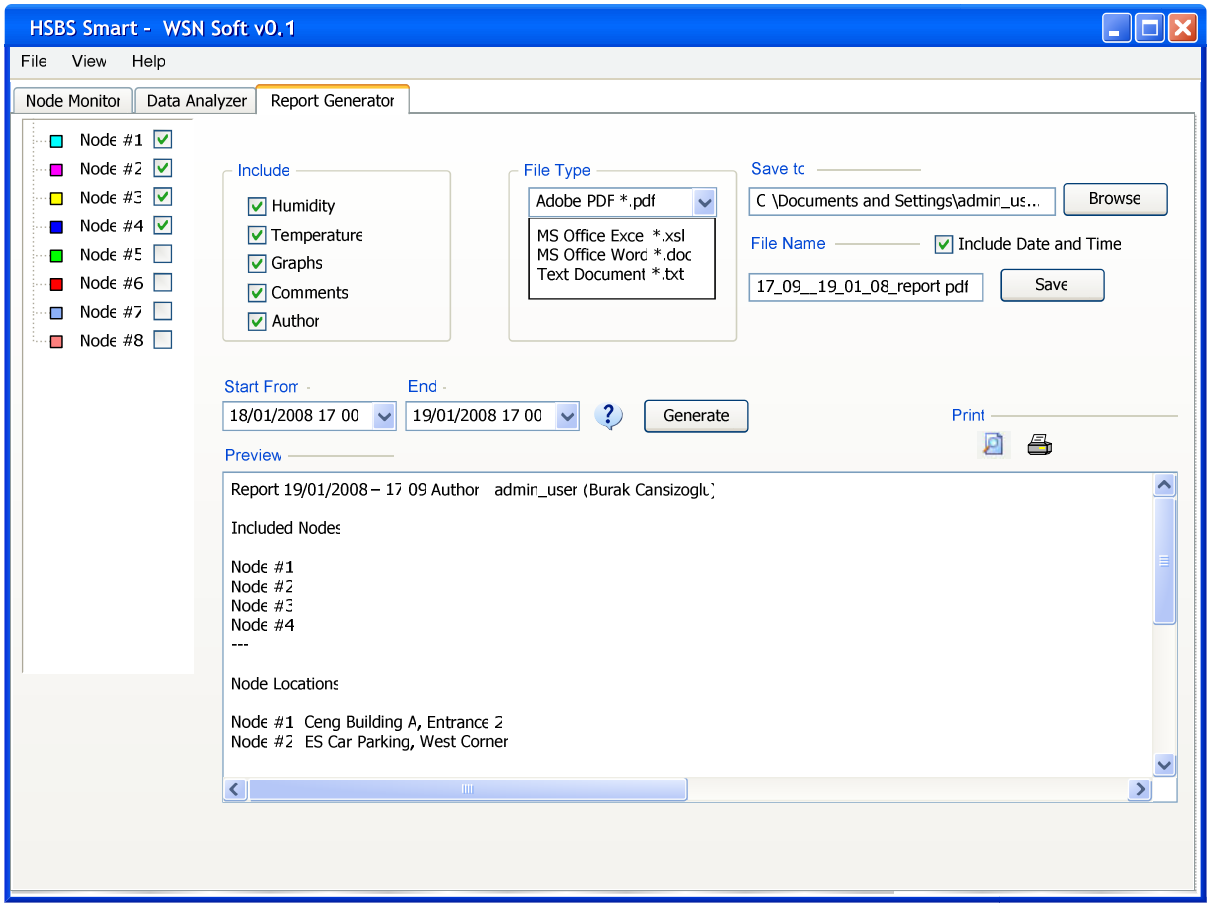


Figure 5-7 Report Generator

Report Generator in tab in Figure 5-7 contains seven parts as left side node list panel, include option panel, file save panel, generate report panel, time panel, print panel and preview panel.

The left side node list panel is same with the panel in Data Analyzer tab and lets the user by check boxes to define to include or exclude the nodes which the report will be created about.

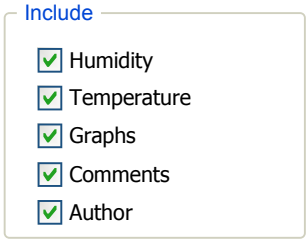


Figure 5-8 Include Panel

Include options panel allows user to select the information which is going to be reported. User can leave out the unwanted info by deselecting the checkboxes on each row. The include options panel can be seen in detail in Figure 5-8.

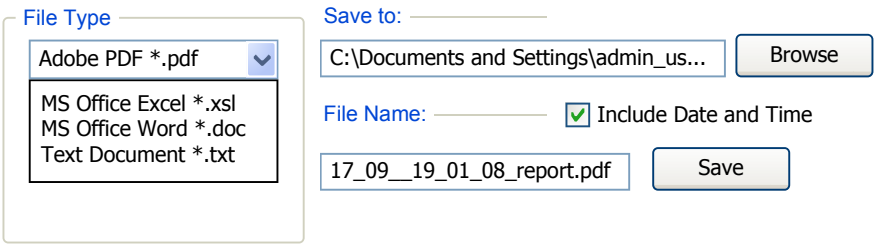


Figure 5-9 File Save Panel

The file save panel in Figure 5-9 has two parts; first one is file type selection combo box and second part is path and file name textbox and buttons. The user can select through four types of file formats which are Adobe PDF, MS Office Excel, MS Office Word and plain text document. Save to text input box displays the path where the file to be saved and file name text input file allows user to type in a custom name. The user can check the "include date and time" check box allowing program to add time and date info to file name.

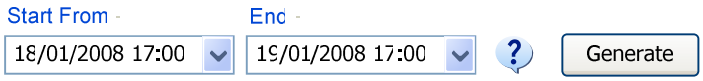


Figure 5-10 Generate Report Panel

Generate report panel consists of a time panel, a help icon and a generate button which are used for defining the time interval that the information in the report will be built on, displaying a calendar for supporting the user while deciding on interval and generating the report accordingly.

Figure 5-11 Print Panel

Print panel has two icons representing the print preview and print functions. User can preview the generated report before printing by clicking on the print preview icon and can print the report by clicking the print icon.

The preview panel is to display the preview of the generated report in a simple text view mode. Objects like graphs or images are included as tags, (<<graph01>>, <<image002>>).

6 PROJECT SCHEDULE

6.1 Finished Work

- We have compiled Embedded Linux Kernel on AP-400 with Pardus.
- We could reach Linux that operates on AP-400 by a helper tool which is CuteCom. We have tried some of the commands which are on the "MadWifi/Atheros Wireless Linux Driver Users Guide" by using CuteCom. We are now able to communicate with AP-400 via a PC.
- We have been provided a CENG 336 Embedded Board and in order to get familiar with the PIC 16F877 we have programmed it with the codes which were written in the CENG 336 course last year. Moreover, we have studied the PIC 16F877 datasheet.
- We have obtained the CCS C Compiler and a recitation about it has been carried out by our assistant. Furthermore, we wrote some codes on this tool and compiled them successfully.
- We have acquired WinPic800 which is a helper program. This program is used to send the '.hex' files which were formed after compilation of the codes on CCS C Compiler, to the PIC 16F877 via a parallel cable.
- The SHT15 Temperature & Humidity Sensor Datasheet has been examined. Furthermore, the schematic of SHT15 has been analyzed.
- Inter-Integrated Circuit Bus Protocol (I^2C) which is going to be used in data transferring between SHT15 and PIC 16F877 has been investigated. How this protocol works has been researched.
- RS-232 Serial Communication Bus Protocol which is going to be used in data transferring between AP-400 and PIC 16F877 has been examined.
- We have been studying "Embedded Linux System Design and Development" book.
- "MadWifi/Atheros Wireless Linux Driver Users Guide" which explains the wireless

Linux commands has been studied.

6.2 Future Work

- We will continue to explore AP-400 on the following days.
- We will study “Atheros AR2317 Processor” datasheet.
- We will peruse “ADM Tech ADM6996F Single Chip Ethernet Switch Controller” datasheet.
- We will read “SpiFlash NX25P16 Serial Flash Memory” datasheet.
- We will study “Altium Designer” which is going to be used to draw the PCB.
- We will study on IEEE 802.11 protocol.
- We will design the schematic of the PCB and then draw it using “Altium Designer”. After this, the drawing of the PCB will be printed, and then the PCB will be tested.
- We will program the PIC 16F877 in order to transfer data from SHT15 to PIC 16F877 via I^2C protocol.
- After getting data from SHT15 via I^2C protocol, we will program the PIC 16F877 in order to transfer data from PIC 16F877 to AP-400 by RS-232 protocol.
- We will adjust AP-400 and after adjusting, AP-400 will be able to receive data that is coming from PIC 16F877.
- Next, we will write a program that will run on AP-400 and this program will send the data that is gathered from sensors to the server.
- After above steps have been accomplished, we will obtain an HSBS Sentinel and it will be tested.
- The next step is to implement a server that will receive data sent from HSBS Sentinels, store the data in a database. The server will also be used to monitor and query the data in the database via a user interface.

- The last step is to build a wireless mesh network via a number of HSBS Sentinels and a server.

6.3 Gantt Chart

Gantt Charts of first and second semester can be found in Appendix-A and Appendix-B accordingly.

7 REFERENCES

1. Wikipedia, Wireless Sensor Network, October 2007, http://en.wikipedia.org/wiki/Wsn

2. Jason Lester Hill, System Architecture for Wireless Sensor Networks, 2003, http://www.jlhlabs.com/jhill_cs/jhill_thesis.pdf

3. SHT1x Breakout Board, 2007, http://www.sparkfun.com/datasheets/Sensors/ SHT15-Breakout-Schematic.pdf

4. P. Raghavan, Amol Lad, Sriram Neelakandan, “Embedded Linux System Design and Development”, Auerbach, (2005).

APPENDICES

Appendix A

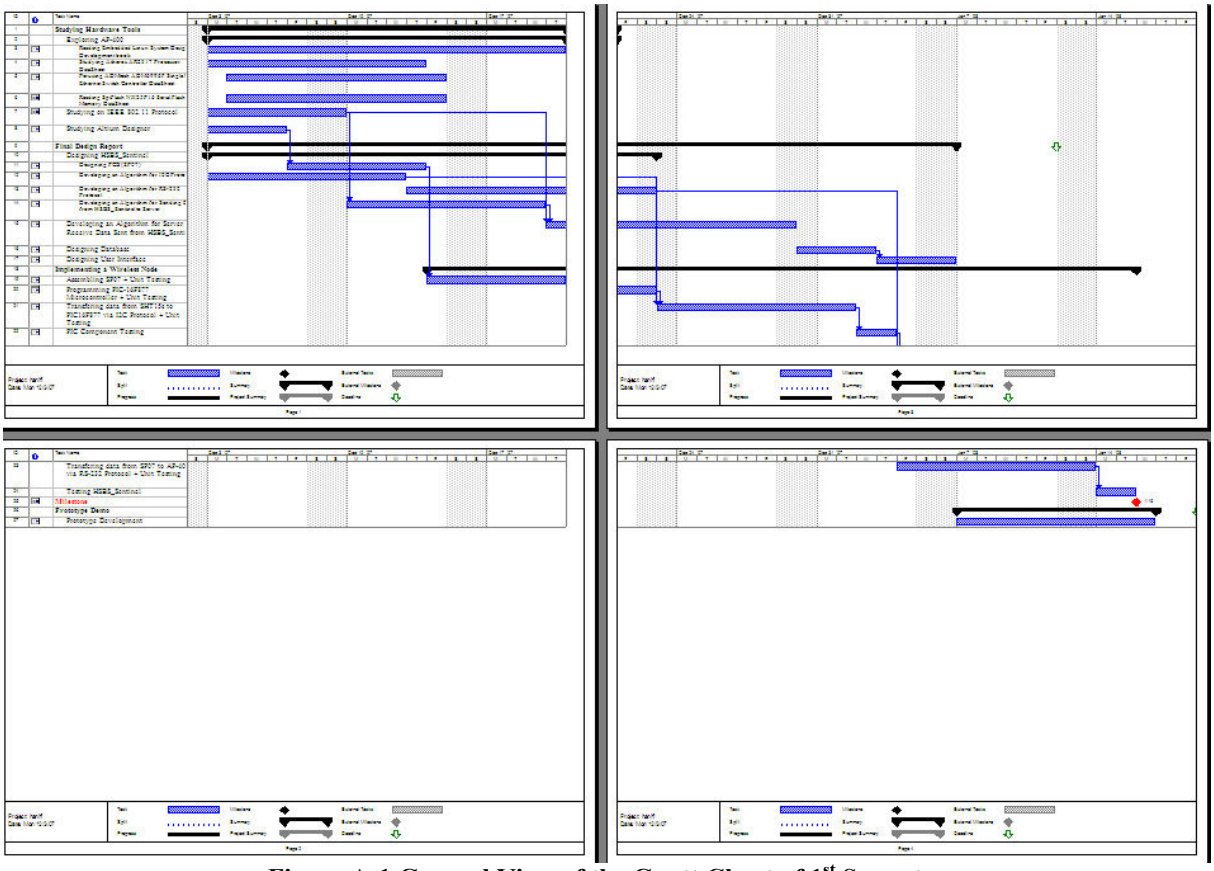
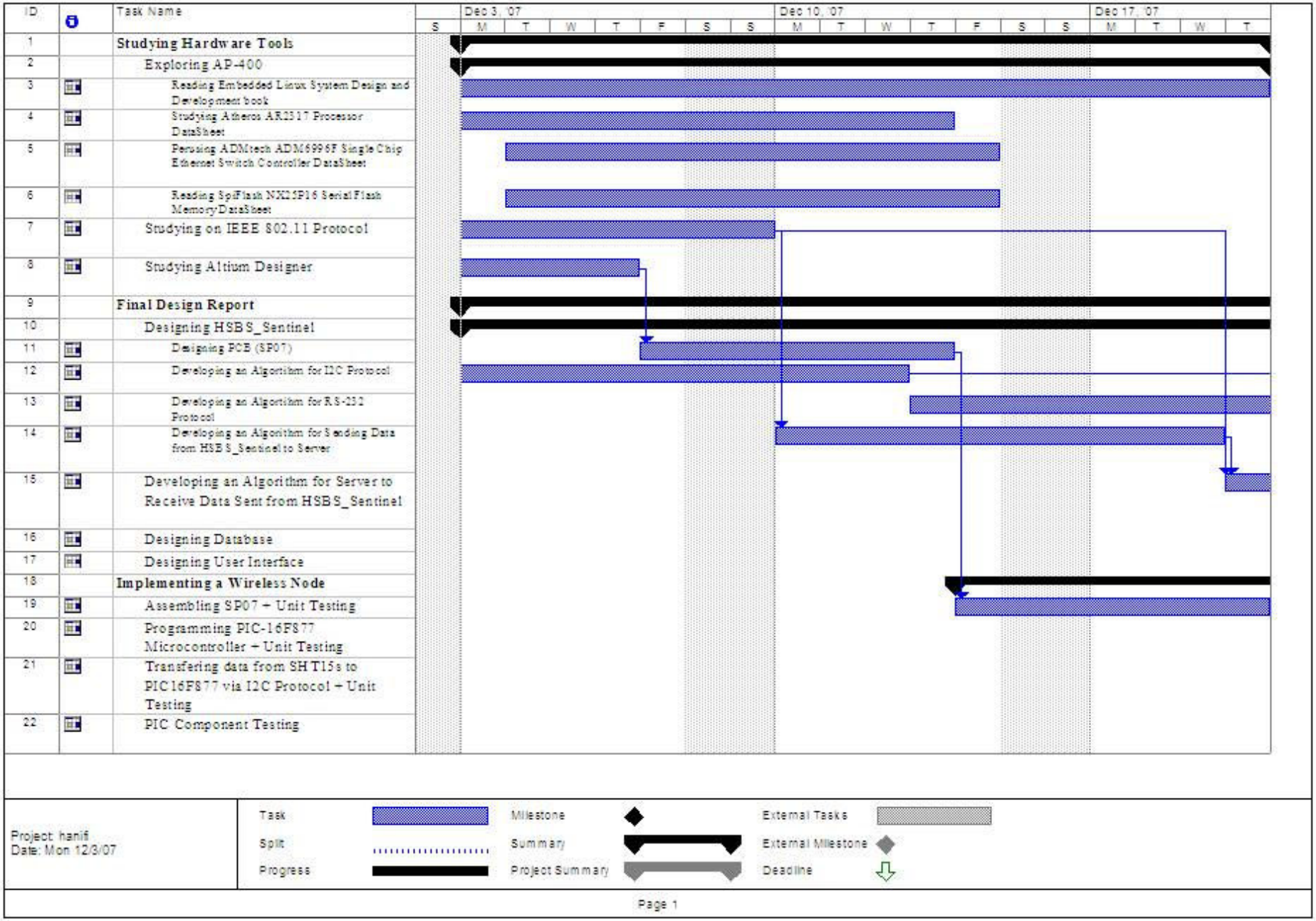
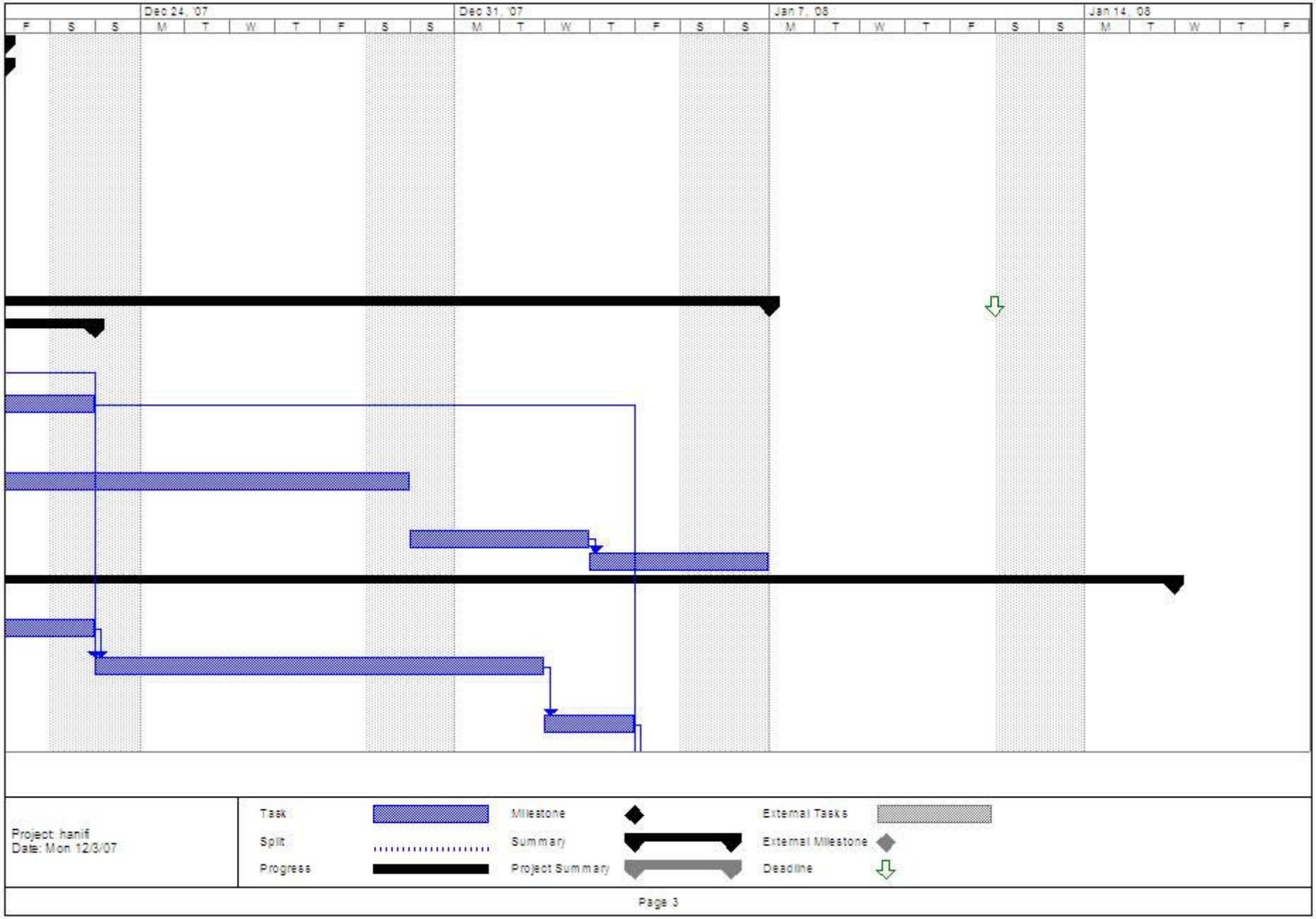


Figure A-1 General View of the Gantt Chart of 1st Semester

Following 4 images show the Gantt Chart of 1st Semester in detail.

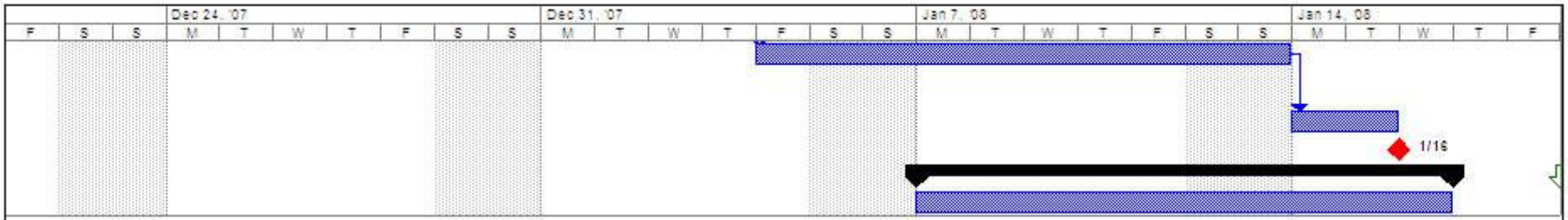

ID
Task Name
Dec 3, '07
Dec 10, '07
Dec 17, '07
S M T W T F S S M T W T F S S M T W T
1 Studying Hardware Tools
2 Exploring AP-400
3 Reading Embedded Linux System Design and Development book
4 Studying Atheros AR2317 Processor DataSheet
5 Perusing ADMtech ADM6996F Single Chip Ethernet Switch Controller DataSheet
6 Reading SpiFlash NX25P16 Serial Flash Memory DataSheet
7 Studying on IEEE 802.11 Protocol
8 Studying Altium Designer
9 Final Design Report
10 Designing HSBS_Sentinel
11 Designing PCB (SP07)
12 Developing an Algorithm for I2C Protocol
13 Developing an Algorithm for RS-232 Protocol
14 Developing an Algorithm for Sending Data from HSBS_Sentinel to Server
15 Developing an Algorithm for Server to Receive Data Sent from HSBS_Sentinel
16 Designing Database
17 Designing User Interface
18 Implementing a Wireless Node
19 Assembling SP07 – Unit Testing
20 Programming PIC-16F877 Microcontroller – Unit Testing
21 Transfering data from SHT15s to PIC16F877 via I2C Protocol – Unit Testing
22 PIC Component Testing
Project: hanif
Date: Mon 12/3/07
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 1


Dec 24, '07
Dec 31, '07
Jan 7, '08
Jan 14, '08
F S S M T W T F S S M T W T F S S M T W T F S S M T W T F
Project: hanif
Date: Mon 12/3/07
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 3

ID	Task Name	Dec 3, '07							Dec 10, '07							Dec 17, '07			
		S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T
23	Transfering data from SP07 to AP-400 via RS-232 Protocol – Unit Testing																			
24	Testing HSBS_Sentinel																			
25	Milestone																			
26	**Prototype Demo**																			
27	Prototype Development																			

Project: hanif
Date: Mon 12/3/07

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline



40

Project: hanif
Date: Mon 12/3/07

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

Appendix B

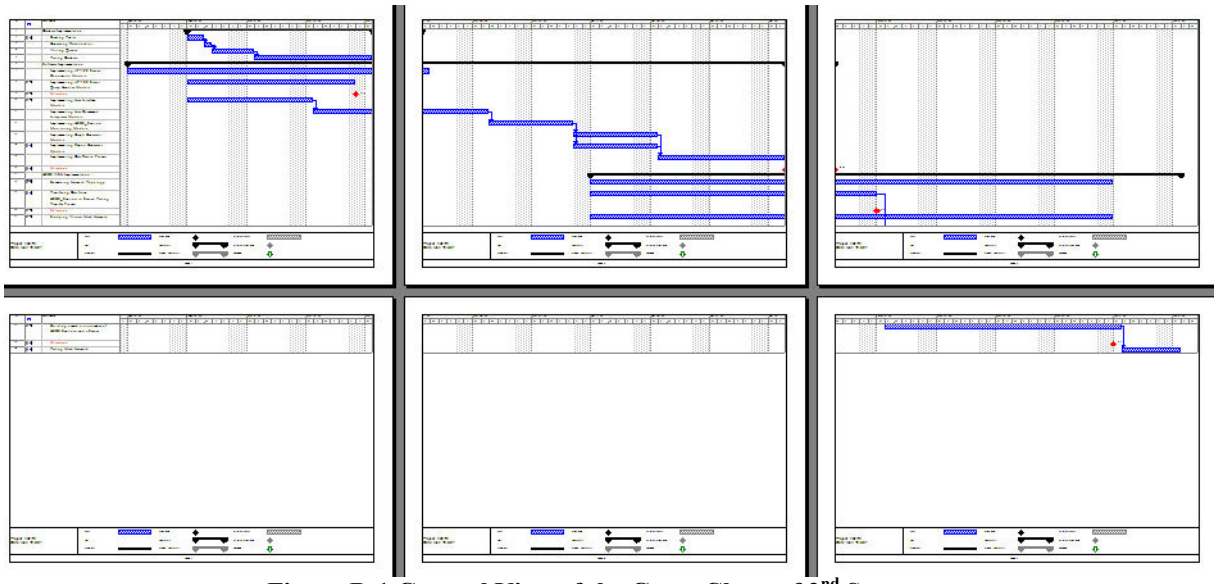
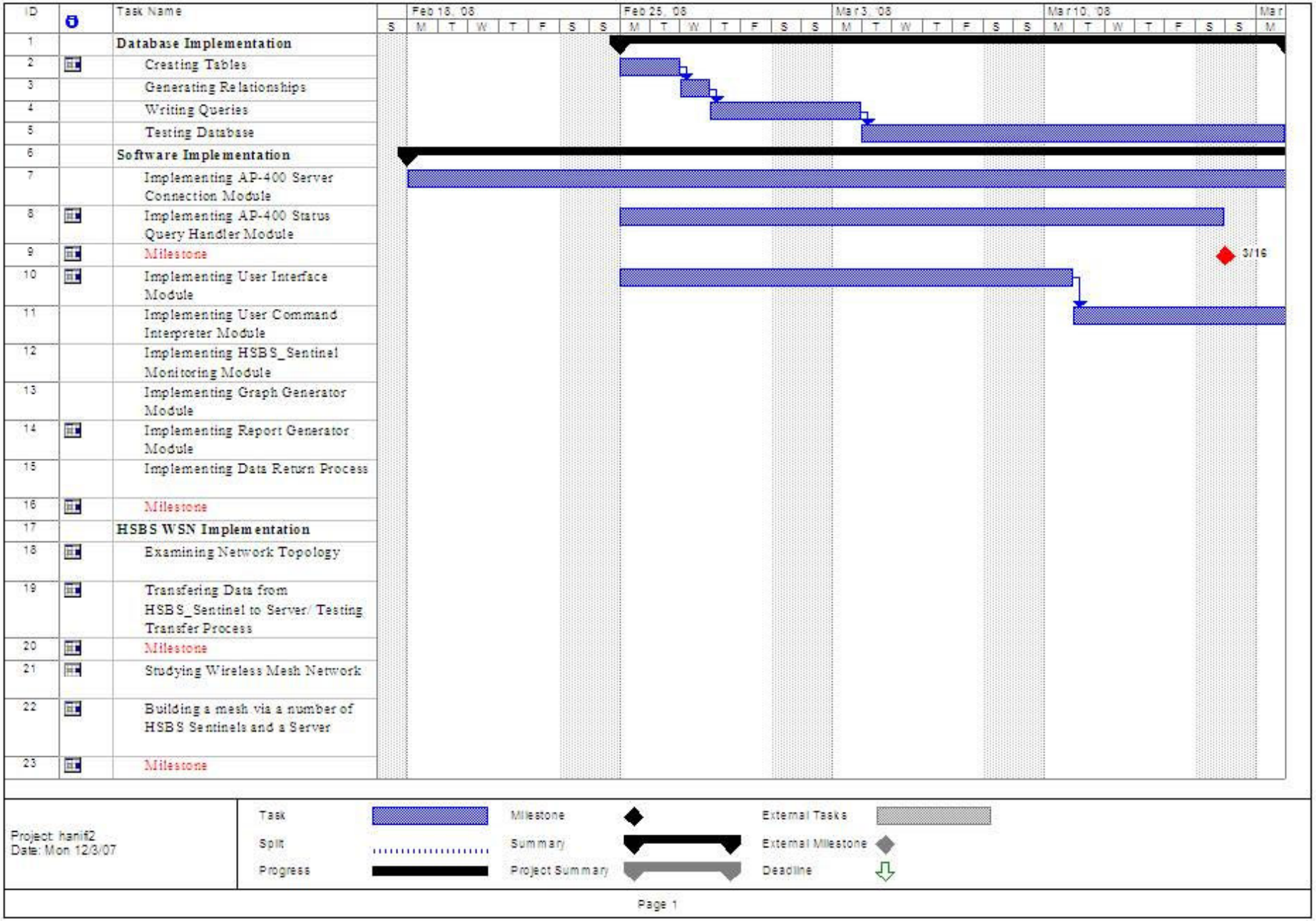
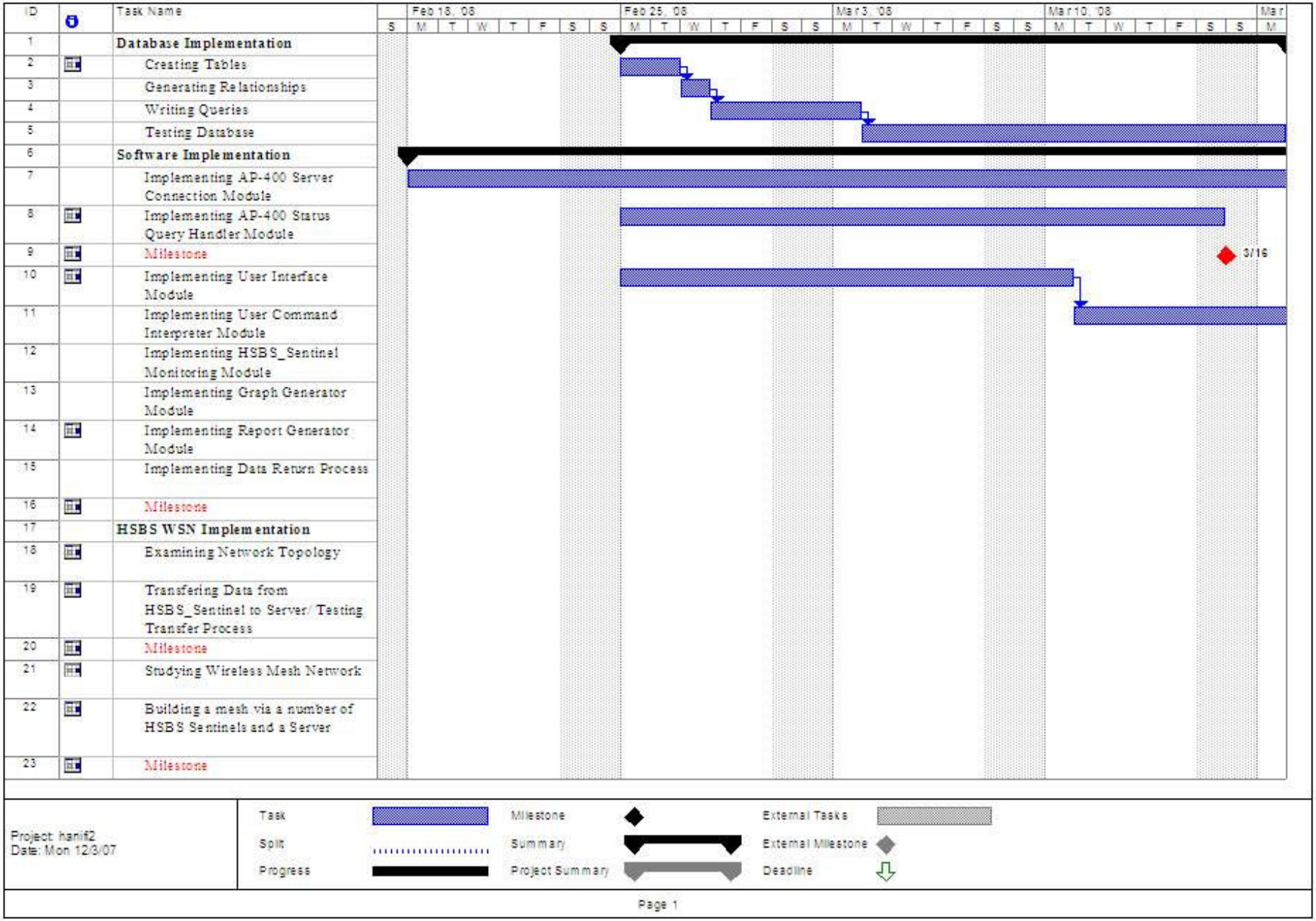
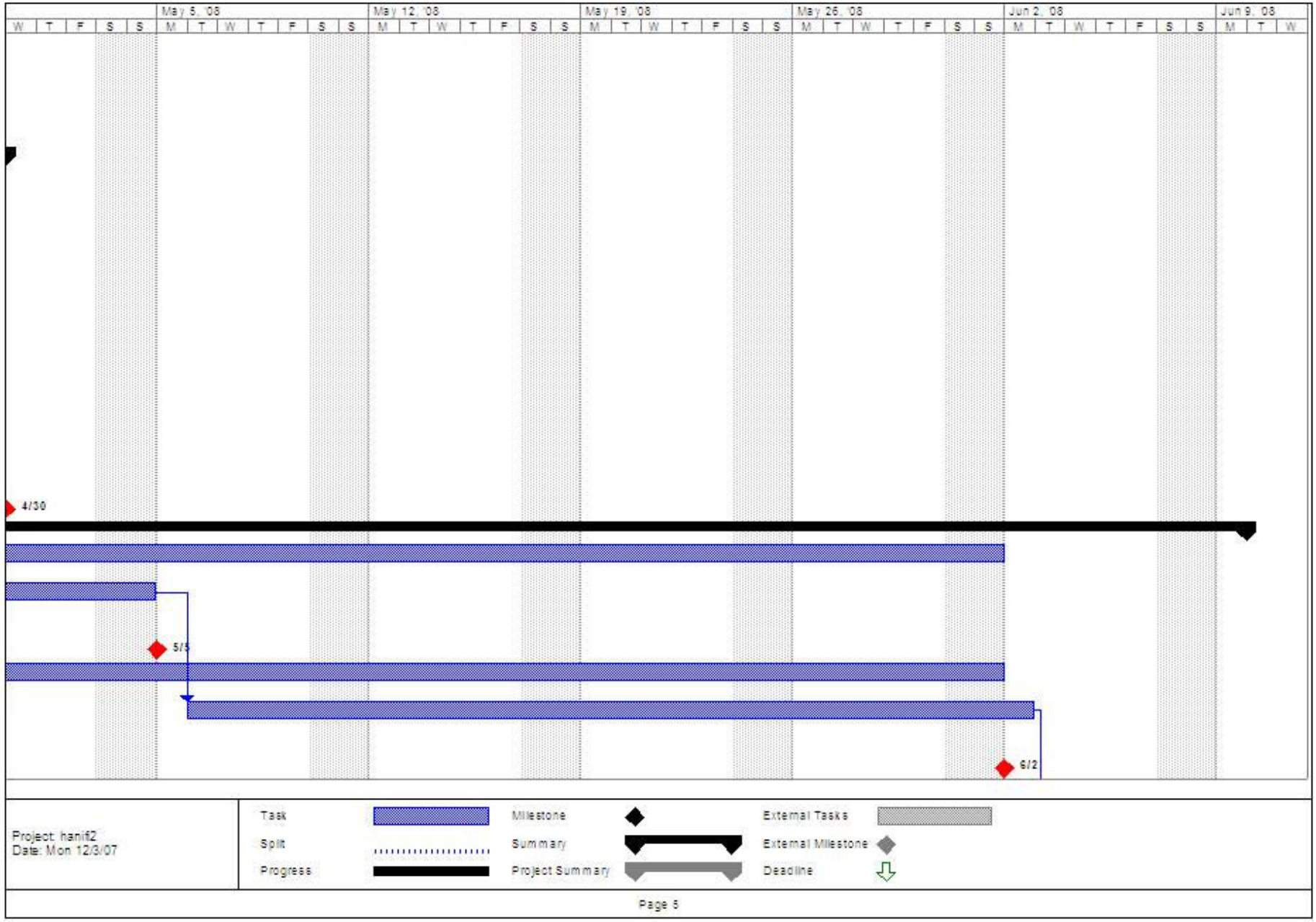


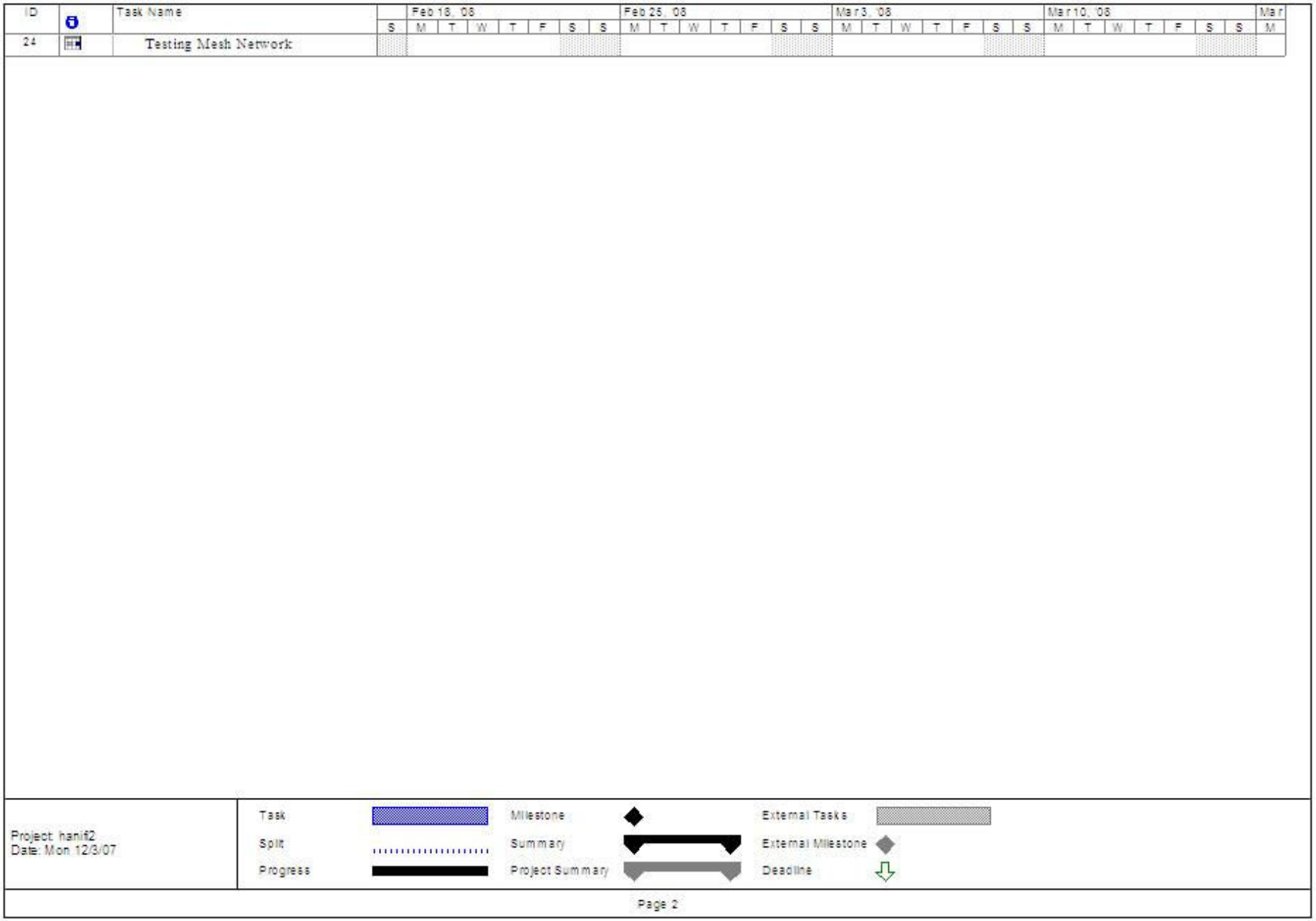
Figure B-1 General View of the Gantt Chart of 2nd Semester

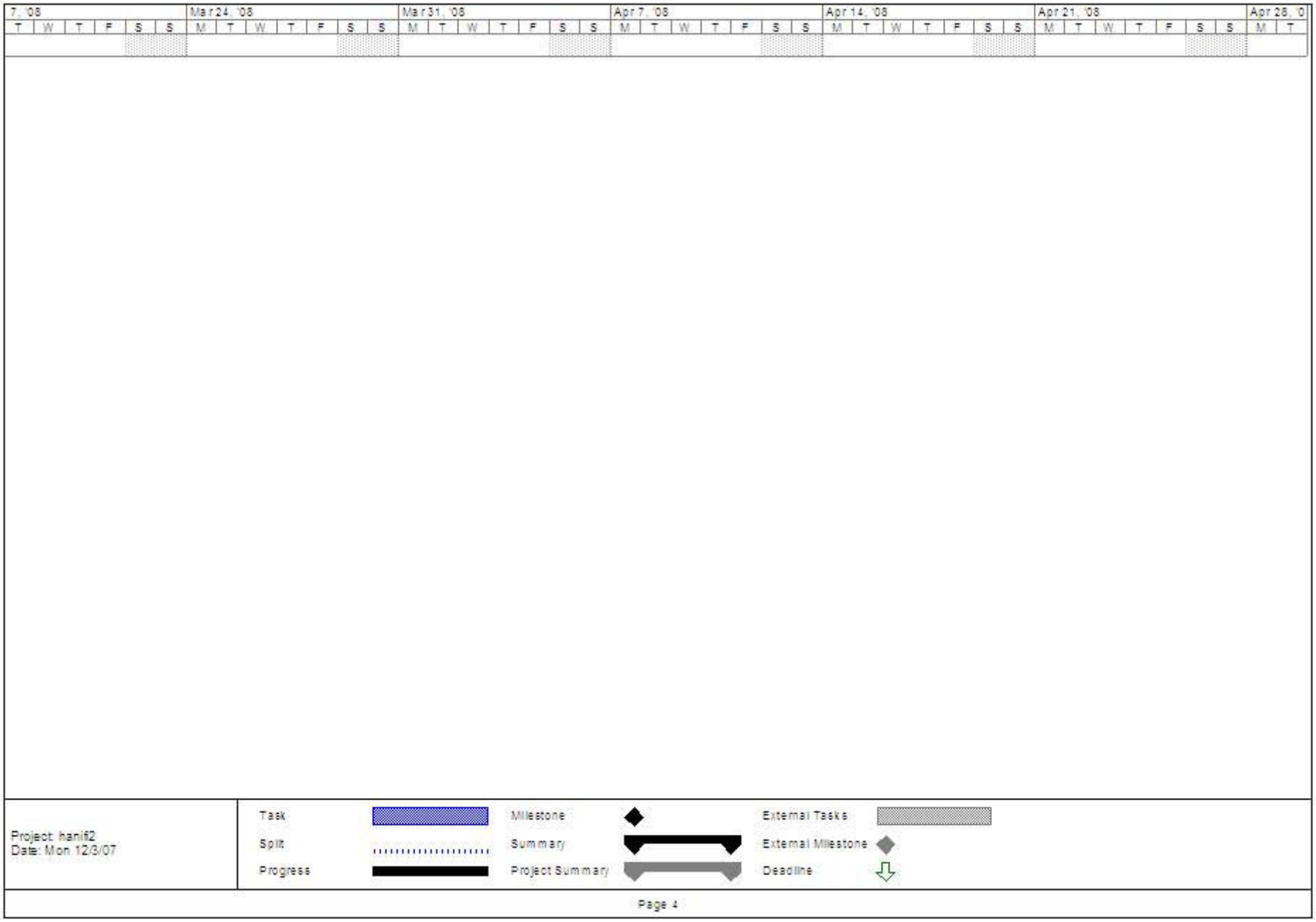
Following 6 images show the Gantt Chart of 2nd Semester in detail.

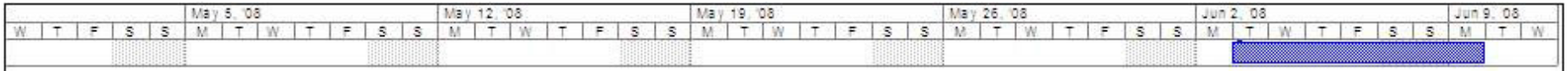

ID
Task Name
Feb 18, '08
Feb 25, '08
Mar 3, '08
Mar 10, '08
Mar
S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M
1 Database Implementation
2 Creating Tables
3 Generating Relationships
4 Writing Queries
5 Testing Database
6 Software Implementation
7 Implementing AP-400 Server Connection Module
8 Implementing AP-400 Status Query Handler Module
9 Milestone
3/16
10 Implementing User Interface Module
11 Implementing User Command Interpreter Module
12 Implementing HSBS_Sentinel Monitoring Module
13 Implementing Graph Generator Module
14 Implementing Report Generator Module
15 Implementing Data Return Process
16 Milestone
17 HSBS WSN Implementation
18 Examining Network Topology
19 Transfering Data from HSBS_Sentinel to Server/ Testing Transfer Process
20 Milestone
21 Studying Wireless Mesh Network
22 Building a mesh via a number of HSBS Sentinels and a Server
23 Milestone
Project: hanif2
Date: Mon 12/3/07
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 1


ID
Task Name
Feb 18, '08
Feb 25, '08
Mar 3, '08
Mar 10, '08
Mar
S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M
1 Database Implementation
2 Creating Tables
3 Generating Relationships
4 Writing Queries
5 Testing Database
6 Software Implementation
7 Implementing AP-400 Server Connection Module
8 Implementing AP-400 Status Query Handler Module
9 Milestone
3/16
10 Implementing User Interface Module
11 Implementing User Command Interpreter Module
12 Implementing HSBS_Sentinel Monitoring Module
13 Implementing Graph Generator Module
14 Implementing Report Generator Module
15 Implementing Data Return Process
16 Milestone
17 HSBS WSN Implementation
18 Examining Network Topology
19 Transfering Data from HSBS_Sentinel to Server/ Testing Transfer Process
20 Milestone
21 Studying Wireless Mesh Network
22 Building a mesh via a number of HSBS Sentinels and a Server
23 Milestone
Project: hanif2
Date: Mon 12/3/07
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 1


May 5, '08
May 12, '08
May 19, '08
May 26, '08
Jun 2, '08
Jun 9, '08
W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W
4/30
5/5
6/2
Project: hanif2
Date: Mon 12/3/07
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 5


ID
Task Name
Feb 18, '08
Feb 25, '08
Mar 3, '08
Mar 10, '08
S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M
24
Testing Mesh Network
Project: hanif2
Date: Mon 12/3/07
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 2


7, '08
Mar 24, '08
Mar 31, '08
Apr 7, '08
Apr 14, '08
Apr 21, '08
Apr 28, '0
T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T
Project: hanifi2
Date: Mon 12/3/07
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 4



Project: hanif2
Date: Mon 12/3/07

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline