Method Declarations of the Project

Pic Module:

sht15.c:
void comstart (void): This function alerts SHT15.

int1 comwrite (int8 iobyte): This function writes data to SHT15.

int16 comread (void): This function reads data from SHT15.

void comwait (void): This function waits for SHT15 reading.

void comreset (void): This function resets SHT15 communication.

void sht_soft_reset (void): This function resets SHT15(related software).

int16 measuretemp (void): This function measures SHT15 temperature.

int16 measurehumid (void): This function measures SHT15 humidity.

void calculate_data (int16 temp, int16 humid, float & tc, float & rhlin, float & rhtrue): This function calculates SHT15 temperature & humidity.

void sht_rd (float & temp, float & truehumid): This function measures(shift right measured data) & calculates SHT75 temperature & humidity.

void sht_init (void): This function initialises SHT15 on power-up.

void receiveData(): This function sends&receives data from RS232.

ds18b20.c:
void onewire_reset(): This function is OK if just using a single permanently connected device.

void onewire_write(int data): This function writes a byte to the sensor. Parameters are byte - byte to be written to the one-wire.
int onewire_read(): This function reads the 8-bit (1 byte) data via the one-wire sensor and returns 1 byte data from sensor.

float ds1820_read(): This function reads data from DS18b20 and calculates the data. If resolution is 0.1 C degree then the result divides 16.0.

void receiveData(): This function sends & receives data from RS232.

**GUI Module:**

dbConnection.java:

public dbConnection(): Creates a connection with the database server.

public ResultSet returnQueryResult(String query): Executes the “Select” queries.

returnQueryExecute(String query): Executes the “Insert” and “Update” queries.

NodeMonitor.java:

public NodeMonitor(): Extracts sensor info from the database. Checks whether nodes are active.

private void nodeTreeMouseReleased(java.awt.event.MouseEvent evt): Shows the menu when right-clicked to Nodes.

private void checkActivity(): Shows the active nodes as green and inactive ones as red.

private void pingMenuItemActionPerformed(java.awt.event.ActionEvent evt): When right-clicked to a node, it performs a ping action to this node and writes the result of the operation to the log panel.

private void rebootMenuItemActionPerformed(java.awt.event.ActionEvent evt): When right-clicked to a node, it sends a special string to the AP-Module of that node and make that node reboot.

private class MyRenderer: Renders the NodeTree in an iconized structure.
DataAnalyzer.java:
public DataAnalyzer(): Extracts the sensors information from the database. Assigns a graph
drawing function to each sensor. Gives a different color to each sensor graph. Adds the graph
library and initiates it. Creates temperature and humidity tabs.

private void drawButtonActionPerformed(java.awt.event.ActionEvent evt): Creates a dataset
according to the selected time interval and selected nodes.

private void initGraph(): Does the library specific initialization.

private void initFunctions(java.util.Date fromDate, java.util.Date toDate): Takes a time
interval and extraextracts the related information and creates a dataset accordingly.

private void initTimer(): Initiates the timer.

ReportGenerator.java:
public ReportGenerator(): Assigns dates to time spinners. Extracts sensor information from
the database. Creates the Checkbox Tree

private void browseButtonActionPerformed(java.awt.event.ActionEvent evt): Helps the user
to select a location for file saving.

private void genButtonActionPerformed(java.awt.event.ActionEvent evt): Generates graphs
according to criteria.

private void clearButtonActionPerformed(java.awt.event.ActionEvent evt): Clears the editor
pane.

private void saveButtonActionPerformed(java.awt.event.ActionEvent evt): Helps the user
save the generated information as either a .txt or a .pdf file.

SensorSettings.java:
public void loginPage(): Initiates the login page.

public void setSensorTable(): Extracts the sensor information from the database into the
Sensor Table.

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt): Logs in the user if
the information provided by the user is correct.
private void jTable1MouseClicked(java.awt.event.MouseEvent evt): Shows the information in the selected row in the text fields.


public void clearForm(): Clears the update form.

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt): Updates the Sensors Table in the database.

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt): Deletes a sensor in the Sensors Table accordingly.

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt): Inserts a new sensor to the Sensors Table in the database.

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt): Sends the password of the user to the his/her e-mail extracted from the database.

MainFrame.java: This is the mainframe of the GUI Module. It adds sub-part objects GUI.

Server Module:

HSBSserverSMS.c:
void init_tty(char* device, int baudrate): Initializes the serial port to communicate with mobile phone.

void recordString(char* value, MYSQL *mysql): Records the data coming from the HSBS_Sentinels to the database. If an alert condition is violated, it sends an SMS to the user for acknowledgement.

void getIp(MYSQL *mysql): Extracts the IPs that are stored in the Sensors Table in the database.

int main(void): Creates a connection with MySQL Server. Calls getIP() method to extract the sensor IP information and for each sensor, it creates a process. Each process tries to establish a connection with its sensor. If a node is unreachable, its process tries to connect to that node
every 5 seconds. Once it is successful, it starts to get values from that node. If a node becomes unreachable while it was reachable, again its process tries to connect to that every 5 seconds. The user can quit the program via entering “q” to the console.

**AP Module:**

**Capboard_bridge.c:**

*void parser(char* value):* The PIC Module sends measured values to the AP Module in a specific format. This method parses these values accordingly.

*void init_tty(char* device, int baudrate):* Initializes the serial port in order to communicate with PIC Module.

*void init_socket(int port):* Allocates a TCP/IP socket and initializes it and starts to listen to that port.

*void server(int sig):* Accepts a connection with the Server Module and once the Server Module is connected, this method starts to send the values coming from the PIC Module.

*int main(int argc, char** argv):* Calls the init_tty(), init_socket() and server() methods respectively.

**Reboot_module.c:**

*void init_socket(int port):* Allocates a TCP/IP socket and initializes it and starts to listen to that port.

*void server(int sig):* Accepts a connection with the Server Module and waits a reboot request from the GUI Module. If it receives the special string from the GUI Module, then it reboots the device.

*int main(int argc, char** argv):* Calls init_tty() and server() methods respectively.
Tecnical List for SP07

PIC 16F877A

20 MHz Cristal

2 x 22 pF Capacitors

2 X 100nF Capacitors

1 mF Capacitor

Diyot

1 K Resistor

3 x 4.7 K Resistors

7805

Reset Button

1 LED

SHT15 Sensor (SparkFun Electronics)

DS18b20 Sensor (SparkFun Electronics)

USB Programming Interface

Serial & Power Cable.