King Fahd University of Petroleum and Minerals
Computer Engineering Department

COE 400

Interactive Ubiquitous Smart Home for Elderly

(Final Report)

Prepared for:
Dr. Muhammad Raad

By: 

Team

12 June 2007

ACKNOWLEDGMENTS

We would like to give a special thank you to Dr. Raad who supported us during this semester and his efforts. We are also happy that we are part of the computer engineering department at this university. Finally, I would like to thank all of those who participated in making this project a total success.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Technology is growing very fast in the field of computer engineering and computer related sciences, in general. Smart, or context-aware, technology became one of the most important and challenging technologies in the world, starting from mobile phones to cameras, airplanes, cars and almost everything that surrounds the environment. The key characteristic behind it and the most popular one, too, is using sensors and embedded systems.

This is the final report that will discuss the design and implementation of COE 400's project, which is Interactive Ubiquitous Smart Home for Elderly. It will give description of the design requirements, logical design, architectural design, feasibility of implementation and a summary.

I- PROJECT GOAL

The goal of this project is to design and implement an interactive smart home for telemedicine use. It will be a system that will help elder people in living a life depending on their selves without the need of nurses or external help. The system will also interact to the elder's actions and giving him/her helpful services. Regular follow ups and visits from doctors and relieves will be taken into account in designing the smart house.

II- PROJECT SCOPE

Design a complete system with hardware and software components integrated together using context-aware as the basis technology of the project. It can be considered as an embedded system. Logical design must be taken into account including scenarios that are going to be simulated. The smart home will be in a typical house that contains a bedroom, a bathroom, a kitchen and a living room.

III- DESIGN REQUIREMENTS

This part of the proposal includes business goals, technical specifications, required components, technologies and applications.

A- Business Goals:

Smart home for elderly will take advantage of using technology instead of human resources e.g. nurse, doctors, etc. It will get the attention of customers when they understand the whole idea behind it. Also, demand on smart home for elderly will increase with time.

B- Safety:

One of the most important requirements of the design is considering safety measures. There will be emergency cases that have to be treated professionally. Most of the safety cases will be discussed and illustrated in the scenarios part of this proposal.

C- Technical Specifications:

There are both hardware and software requirements for the system to function properly. A main computer is required to save records, statistics and take actions properly. Everything will be connected to this main computer to make it a complete system. The requirements of the system are as follows:

1- Hardware Requirements:

Hardware is the heart of this system. Everything will interact with it in the system to give the required outcomes and desired operation and of course it can be divided into many categories. These categories are as follows:

a- Sensors:

A smart environment needs to detect any action that happens around it. Sensors play one of the major roles in the system. It is the only way that detects action of the elder and records his/her behavior. Because of this reason, sensors must be distributed in the proper places to be able to perform their functions efficiently. Every type of sensors will be mentioned with their function and places:

- Fall Sensors:

Falling sensors will be wearable by the elder to make sure that when he/she falls anywhere in the house, the fall will be detected.

- Pressure Sensors:

Pressure sensors will be used to detect the foot step of the elder and therefore, determine his place. Tow pressure sensors will be used at every door in the house to know which room he is currently in.

- Gas and Smoke Sensors:

The gas sensor will be placed in the kitchen while the smoke sensors will be placed in every room to keep the environment of the house free of gases and smokes by detecting them and taking appropriate actions.

- Light sensors:

Light sensors are needed to turn on or turn off the light whenever needed to reduce the power consumption. It will be placed where the elder is located by the pressure sensors and react according to that.

- Blood Pressure Sensors:
It will detect the blood pressure of the elder and report the result to the main computer located in the house and record some statistics.

- Temperature Sensors:
There will be two types of temperature sensors:

1. One will measure the temperature of each room.
2. The other will be medical sensor to detect his/her body temperature.

-Alarm Circuit:
It contains an alarm that is connected to a sensor. The alarm can be an LED or a voice.

- Medical Sensors:
    - Pulse oximeter:
    The pulse oximeter will give results of the heart beat rate of the person. It will also give details about breathing and send it to the main computer.

    - Blood pressure sensor:
    It will be a wearable sensor that is much more like a circuit or a device to give the blood pressure result of the patient to the doctor and record it in the main computer.

    - Emergency button:
    The emergency button will be carried by the elder all the time to contact the system and, therefore, the doctor outside the house in case of emergencies.

b- Wheel Chair:
A smart wheel chair connected wirelessly to the system. It has most of the medical sensors.

c- Microcontrollers:
Sensors cannot take actions by themselves directly. They have to be connected to microcontrollers to do that. Microcontrollers are one of the most important components of the project. They process signals from various devices e.g. fall sensors, pressure sensors, etc. in a way that makes them useful, then provide the processing output to a third device that will take the proper action for the output which is the main computer. Microcontrollers can be connected to other devices either wired or wirelessly. These two ways will be used depending on the placement of the microcontroller in the house and the role of it, too. Also, the microcontrollers will deal with a middle device that will act as an intermediate connection between it and the database to update the database with the latest actions and results.

d- Displays:

Displays will be used to give the elder important instructions such as the time to take his/her medicine and notifications about visitors. Speakers will be embedded with these displays to give a warning to the elder to make sure that he/she noticed the current event.

e- IP Based Cameras:

To monitor activities of the elder in the house, a camera is needed. To make this camera adapt with the smart environment, it has to be interfaced with the system with proper installation. It will keep track of events and will make it easier to monitor the elder if an emergency occurs.

f- Voice Recognition:

One of the requirements of building a smart house is having devices that will respond to voice commands. Voice recognition devices will be used in this project. It will help the elder in giving services and taking actions by just saying the command. Also, it will be very helpful for emergency cases where he/she cannot move, calling for help by saying it will be useful.

2- Software Requirements:

An embedded system is never complete without having software included in the process. Software will organize the functionality of hardware and make it reasonable to understand. The following will be taken into account when designing the software system part:

a- Web-based Application:

With the introduction of ASP.NET, developers have been giving a powerful new set of tools to help resolve many problems. ASP.NET uses an object-oriented development paradigm. In practical terms, this means that every page is a class that derives from the System.Web.UI.Page class. This class provides a number of services to the web developer including caching, rendering, response and request access, etc. The following figure illustrates it:

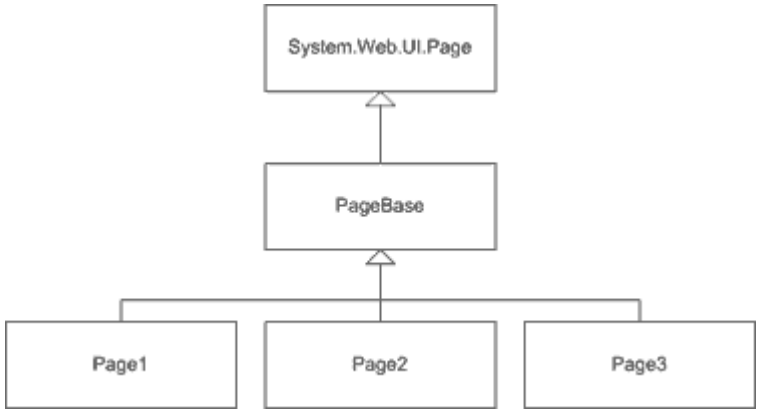


Figure 3.1

- ASP.Net will be used to develop the web-based application of the project.
- ASP.Net will be useful to develop dynamic pages that can interact with the system with many things like
    - Web User Controls
    - Page Inheritance
    - High Performance

This will help constructing solutions for reading data from a Database, writing and updating to it. Additional features of ASP.NET that will help in designing of the software application for the smart home:

1- Wide usage of Microsoft platforms which will work with applications developed by using ASP.NET very efficiently.
2- ASP.NET is simple, modern, and general-purpose application.
3- One of the most popular web programming languages. So sources, tutorials and solution will be easily found when facing problems.

The web application will be connected through the Internet and can be accessed by doctors and relatives. This will be very useful for follow ups and checking for the patient.

b- Database:

Database is required to store information about events that happen in the smart house. It will help Doctors and relatives to keep up with the conditions of the elder.Microsoft SQL Server is a full-featured relational database management system (RDBMS) that offers a variety of administrative tools to ease the burdens of database development, maintenance and administration.There are a number of characteristics of MS SQL that will be very helpful to make the system's web application interactive with it:

1- Easy to install and configure using a simple GUI interface.
2- It comes with neat management tools to keep administration simple and centralized.
3- Rich client/server application can be built efficiently and easily thus reducing cost.
4- Supports communication protocols such as JDBC.
5- The pricing for a MS SQL database is competitive compared to other databases.
6- Secured with password and encryption protection. This will be useful when assigning different access controls for doctors or relatives.
7- Supports backup and recovery of information, which is critical for a system like this to overcome unexpected events such as loss of information in a database or in case of system failure.

c- Client-Server Applications & Communication:

Client-server application will be used to communicate with Ethernet enabled microcontrollers to receive and process data (packets) related to the status of the elderly. It will be implemented using C#.NET.

The communication part concerns about the platforms that are going to be used to send the commands. It will also include the communication technology used e.g. WiFi, Bluetooth, Inferred, etc. It will be done by programming in high level languages. The protocol and the connection of the devices will be defined, too. The following will be taken into consideration when designing the communication part:

- Connection Initiation:
When the system starts running, the clients will request a connection from the server. After the connection is established, the devices can know communicate. The clients will request the connection to the server by using the port number that the server is listening to.

- Protocol:
The protocol is the type messaging used to send the commands between the devices (i.e. microcontrollers) and the server. The protocol type is either a TCP or a UDP. This will depend on the type of the microcontroller used in the project. TCP protocol is going to be used, because it is more reliable than the UDP protocol.

- Medium:
The medium is the type of the technology used for transmitting the commands between the clients and the server i.e. wired or wireless transmission. The connection between the microcontrollers and the PC will be established using the wired technology (Ethernet).

- Wireless Transceivers:
Wireless Transceivers are devices that are used to send and receive the commands from the clients and the server by using wireless technology. Thai can be used for sending commands from the microcontrollers to the server by wireless transmission.

- Communication Language:
It is the High Level Language (HLL) that is used to program the client-server communication. In the project, C# language will be used as mentioned before and by using the VS.net environment.

- Socket Programming:
  Socket Programming is programming between a client and a server. In the socket programming, connection between the client and the server will be initiated. Also, protocol type will be defined using this language. By socket programming we will be able to send and receive the commands from the clients and the server that are connected in the network. Socket programming will be done by using C# language.

Here are some of the specifications of this programming language, which will be efficient for designing the system:

1- Wide usage of Microsoft platforms which will work very efficiently with applications developed by using C#.
2- C# is intended to be a simple, modern, general-purpose, object-oriented programming language.
3- C# is intended to be suitable for writing applications for both hosted and embedded systems.
4- Most popular programming language, so it is easy to find sources, tutorials and solution when facing problems.
5- In C# Microsoft has taken care of C++ problems such as Memory management, pointers etc.
6- The best way to write Windows desktop and web applications.

d- Smart Card:

Smart card will be used in this project in order to enhance the capabilities of the smart home. It will be used by the patient, doctor, and relatives to identify them selves to the smart home. For the patient, it will contain information about his/her medical status to make it easier for the doctor taking treatment and so on. The doctor will be identified also using the smart card kit to take some privileges in the house like, updating the information of the patient, updating access of the visitors or relatives, etc. it will be used by visitors to identify for them selves when accessing the home.

e- Modelling & Algorithms:

Behaviour of the elder and his/her daily routine will be modelled. This will be done depending on the patient's health profile e.g. diabetes, hypertension, etc. The house will be modelled along with the layout of sensors. Artificial intelligence will take part of the modelling design.

IV- TEAMS DIVISION

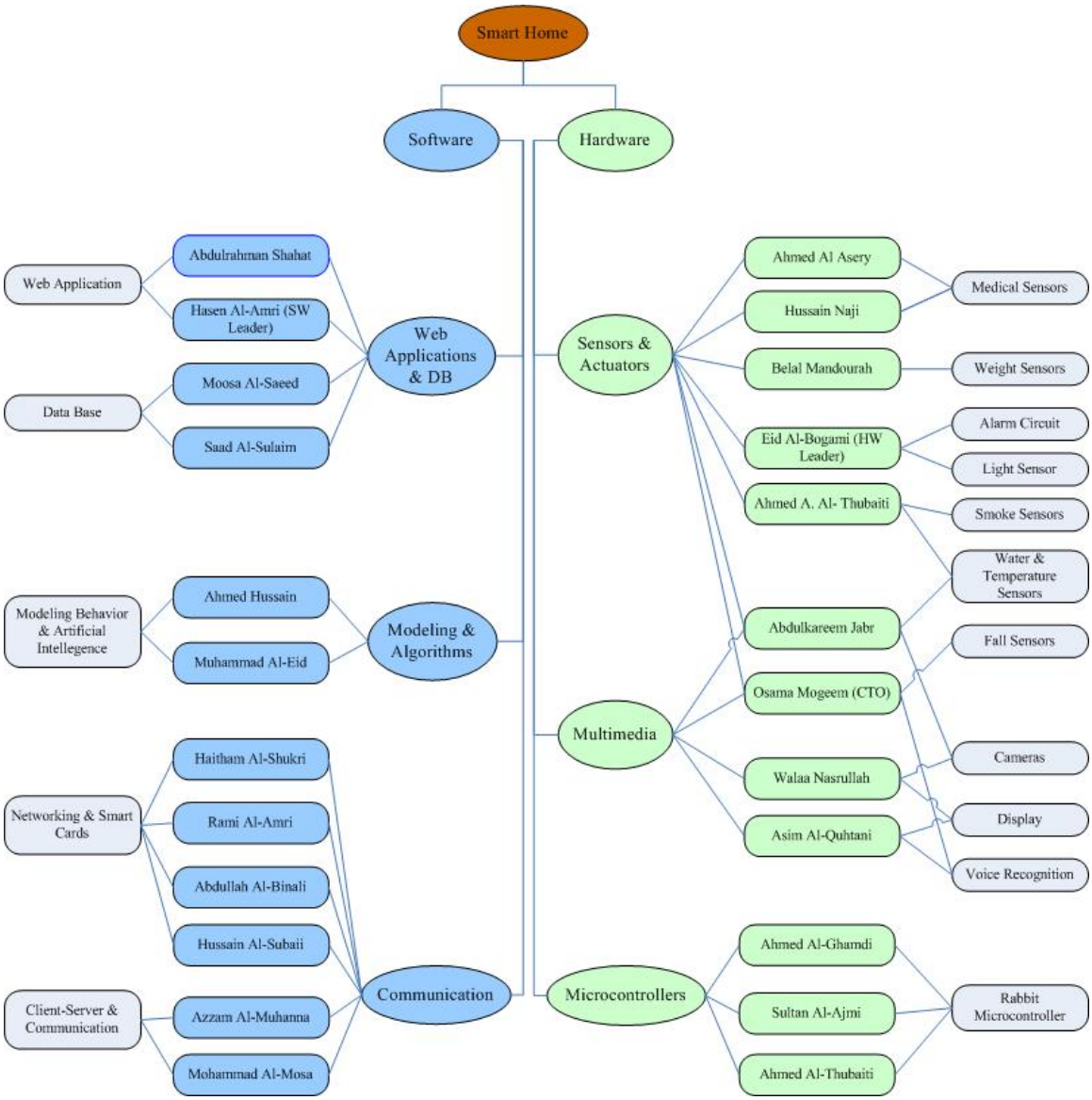


Figure 4.1

V- PROJECT DESIGN

To design a complete system, certain scenarios have to taken into consideration in order to remodel the real problem into a logical problem that can be implemented in both hardware and software. Coming up with feasible, flexile and reliable design is the most important part of building a complete system. Here is the layout of the house, where the system will be implemented:

Figure 5.1

A- Scenarios Modelling:

All scenarios that could happen to the elder will be taken into account. These scenarios are modelled in order to achieve the required results. The proposed scenarios are as follows:

1- Scenario 1 (entering living room):

- When pressure pads inside living room activated, meaning that person is in living room
- Wait for 90 seconds
- If interrupted by another pressure pad then clear timer otherwise check motion detector sensor, if there is motion, then fine. If there is not, it means the person is facing problems and emergency alarm activated.
- If 90 seconds ends without any sensor activated, indicate problem.

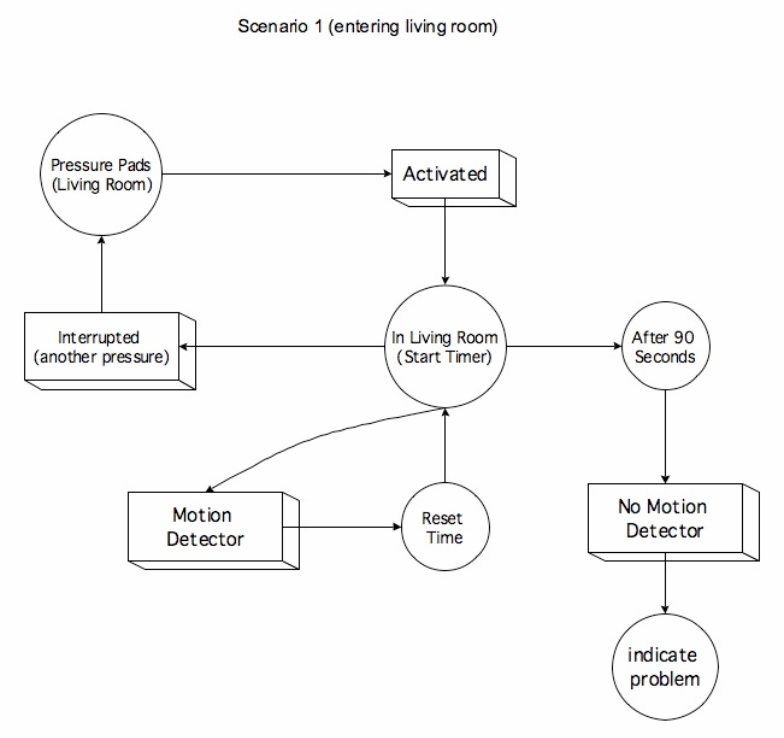


Figure 5.2

2- Scenario 2 (entering the kitchen)

- When the pressure pad inside the kitchen is activated, the person is in the kitchen
- wait for 90 seconds if interrupted by gas sensor from the oven, reset timer, and indicate cooking activity.
- If interrupted by water sensor in the sink, reset timer, and indicate washing activity
- If interrupted by the fridge read switch, reset timer, and indicate food or beverage activity if motion.
- If 90 seconds ends without any sensor activated, indicate problem situation.

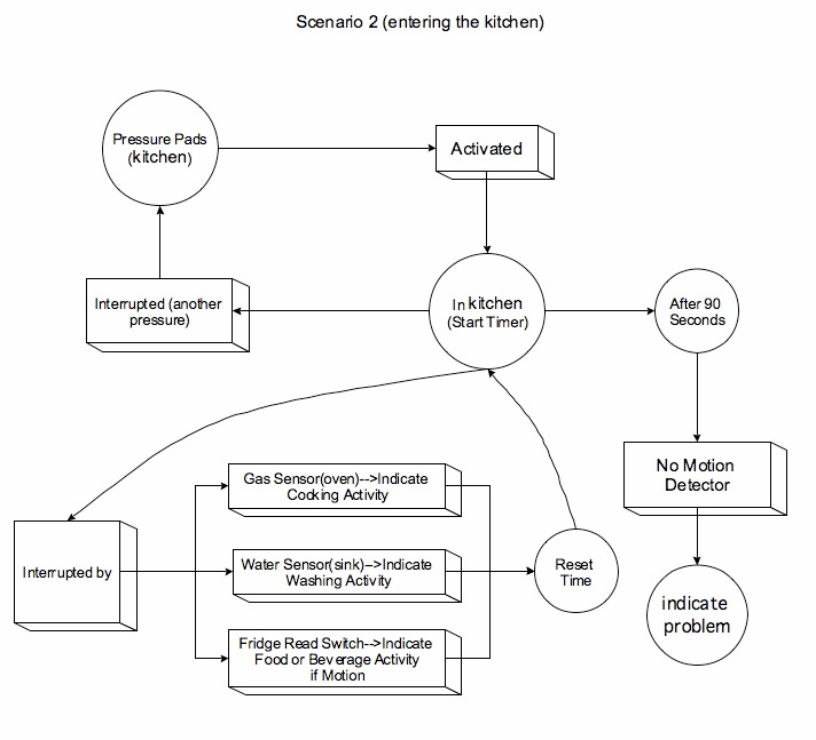


Figure 5.3

3- Scenario 3 (entering the toilet):

Check water sensor of the floor, if activated then light the red light outside the toilet indicating risk of slipping if pressure pad is activated, meaning person entered the toilet

- Wait for 10 minutes reset timer whenever motion detection sensor is activated
- If time ends, indicate problem situation.

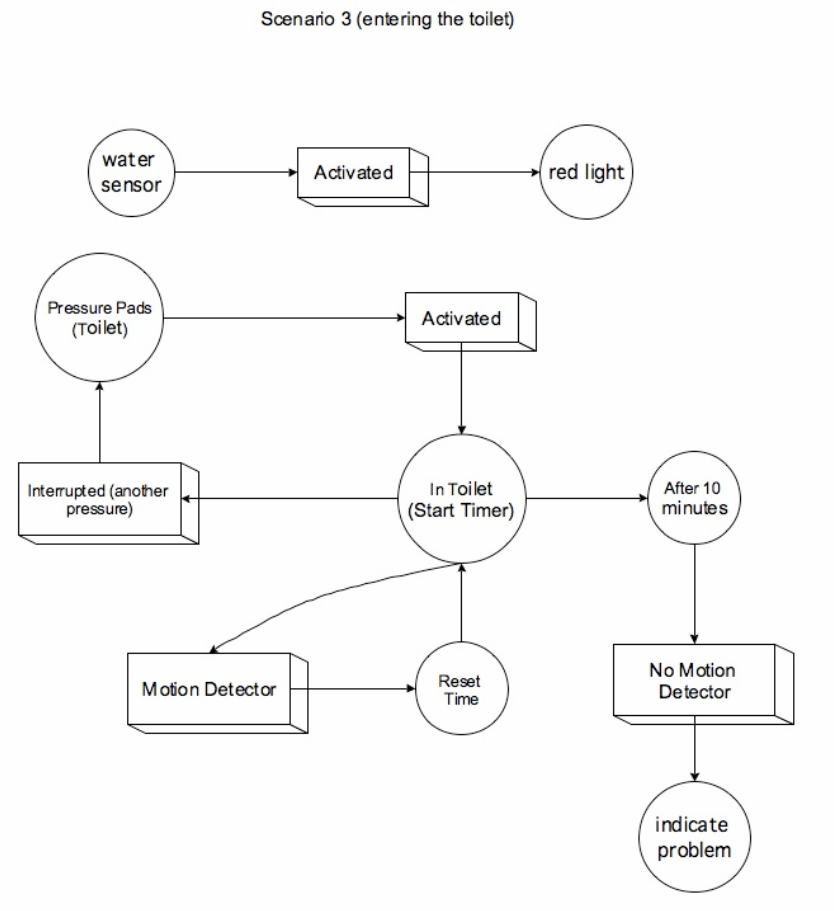


Figure 5.4

4- Scenario 4(sleeping in the bedroom):

If pressure pad inside the bedroom is activated, meaning person inside the bedroom if pressure sensor of the bed is activated, person wants to sleep.

- wait for 10 minute every time check for motion detection sensor and if activated reset timer.
- If 10 minutes end without any activity, person already slept start night alarm sensing around the house and sense for motion detection sensor.
- Start timer for 8 hours after the 8 hours end, wake up voice message and turn on the lights to wake up the elder.

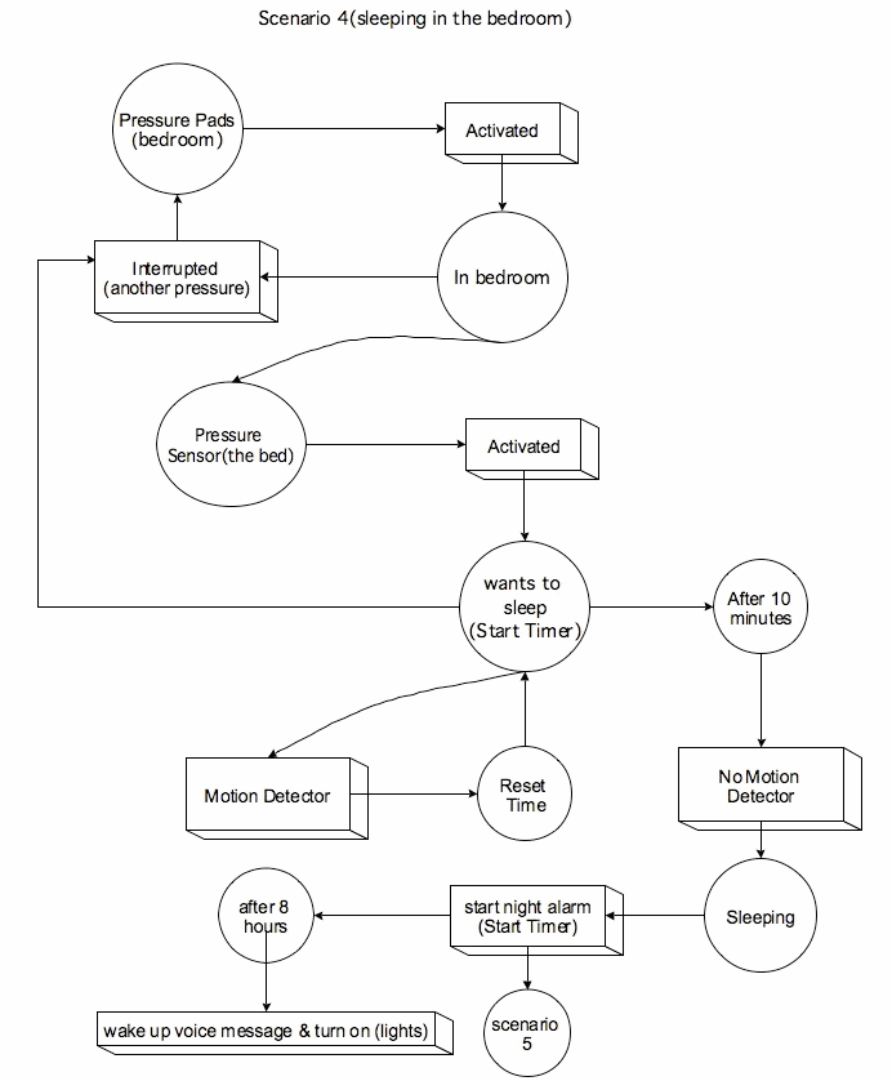


Figure 5.5

5- Scenario 5 (night alarm sensing):

- Sense from all motion detection sensors except the one in the bedroom if any sensor is activated, meaning there is an intruder inside the house.
- start Emergency Alarms, start surveillance cameras, call Police, contact relative and neighbors and alert elder person.

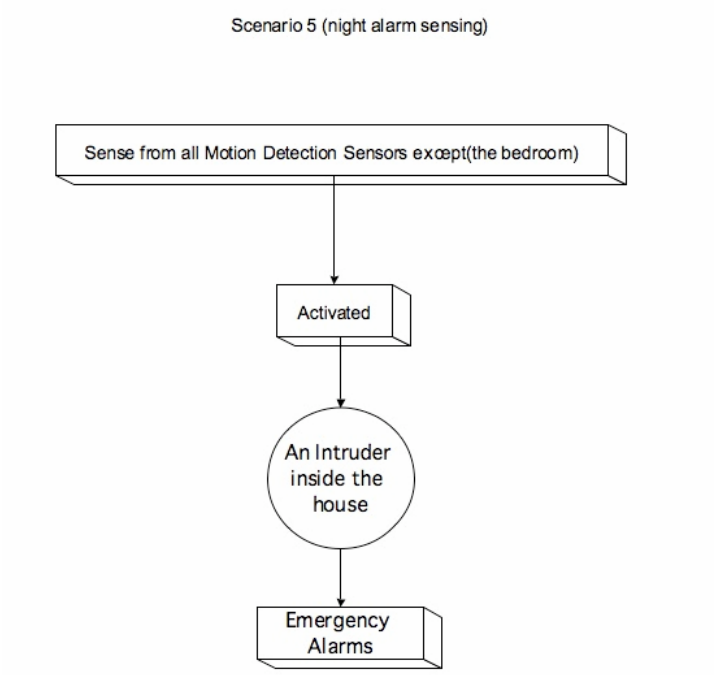


Figure 5.6

6- Scenario 6 (outside the house):

- Sense from all motion detection sensors without any exception if any sensor is activated, meaning there is an intruder inside the house.
- start Emergency Alarms, start surveillance cameras, call Police, contact relative and neighbors and alert elder person.

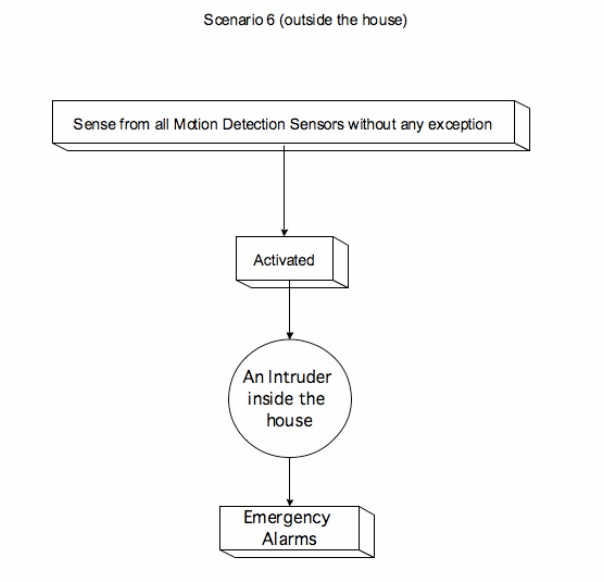


Figure 5.7

7- Scenario 7 (waking up):
- Pressure pad of the bed will be deactivated and motion detection sensor inside the bedroom will be activated.
- stop night alarm sensing.

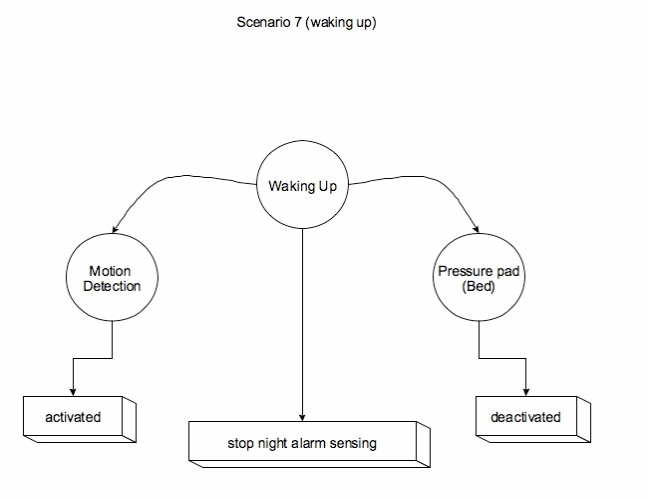


Figure 5.8

8- Scenario 8 (problem indication):
Activate cameras, contact relative for suspicious situation
-wait more 90 seconds if no activity, then call for emergency procedure (call doctor, activate emergency alarm).

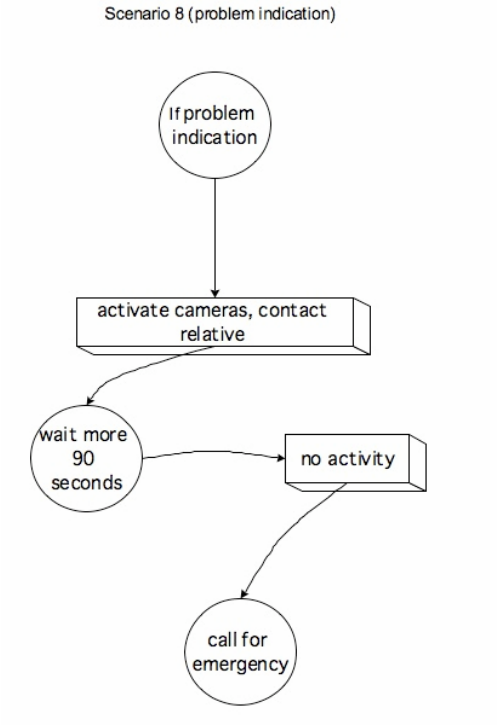


Figure 5.9

9- Scenario 9: Number of times entering the toilet at night:

- After 10 pm, start counter of number of times the elder is entering the toilet
- if counter exceeds 5 times for the whole time, alert doctor or nurse as this means health problems.

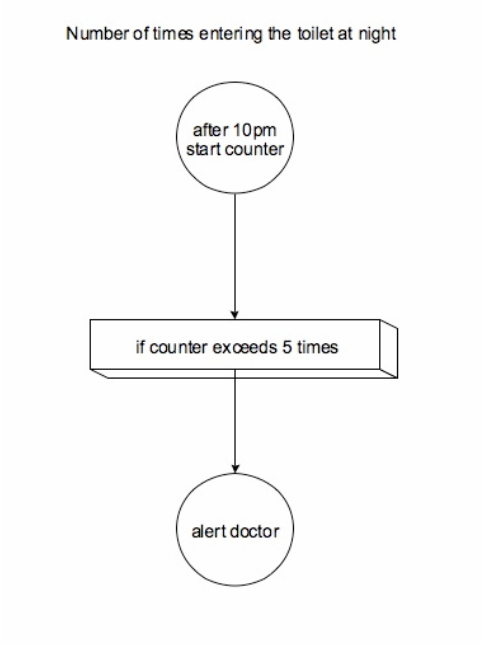


Figure 5.10

10- Scenario 10: Taking medical shots:

The patient should take 3 shots a day and eat something after each one within one hour

- Dispense a shot at 8 am, start the two timers, and lock dispenser
- Timer 1 wait for one hour, if fridge read switch is activated then stop this timer. If one hour ends with no fridge activity, show eating reminder.

-timer 2 wait for 5 hours for the next dispense of the shot.

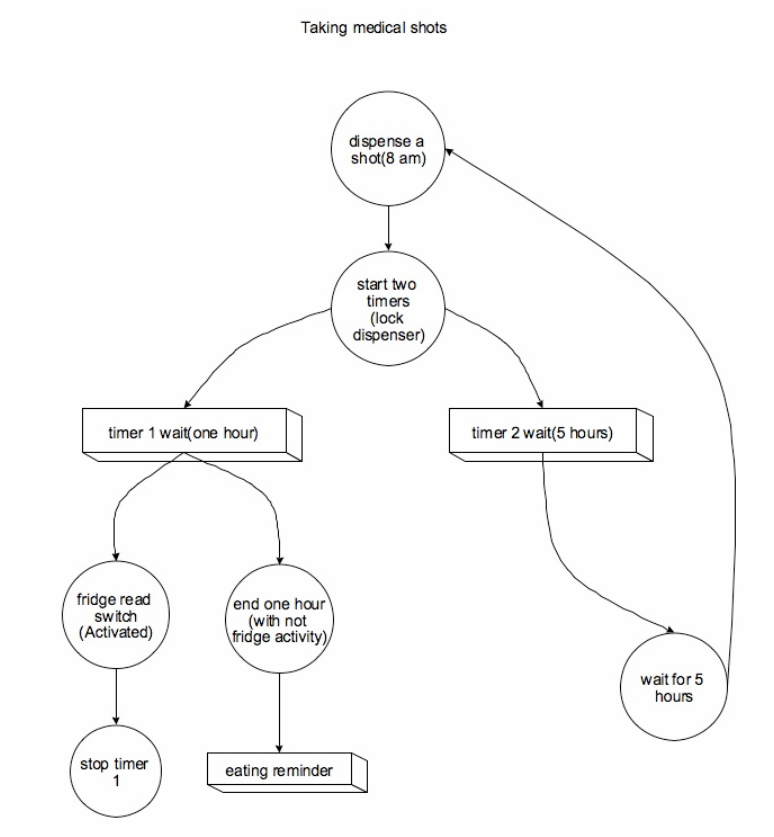


Figure 5.11

Sensors Layout:

Figure 5.12

After the brain storming and a long class discussion especially between the Sensors Group Members, we came out with the above layout of the sensors which we are going to use in the project.

Location	**Sensor**	**Quantity**
Bedroom	Motion Detection	1
	Light Sensor	1
	Smoke Detector	1
	Pressure Pad	1
Bathroom	Water Sensor	1
	Motion Detection	1
	Smoke Detector	1
	Read Switch	1
Living room	Pressure Pad	4
	Smoke Detector	1
	Motion Detection	1
	Light Sensor	1
Kitchen	Water Sensor	1
	Read Switch	1
	Gas Sensor	1
	Pressure Pad	1

B- Software:

1- Database:

The purpose of the database was to design a complete data base that will have all the records of the elder, anything else related to him/her and any action that happens in the smart house. Furthermore, make a connection to the Web and the Client/Server applications. The database was implemented using MS Access. The tables of the database are as follows:

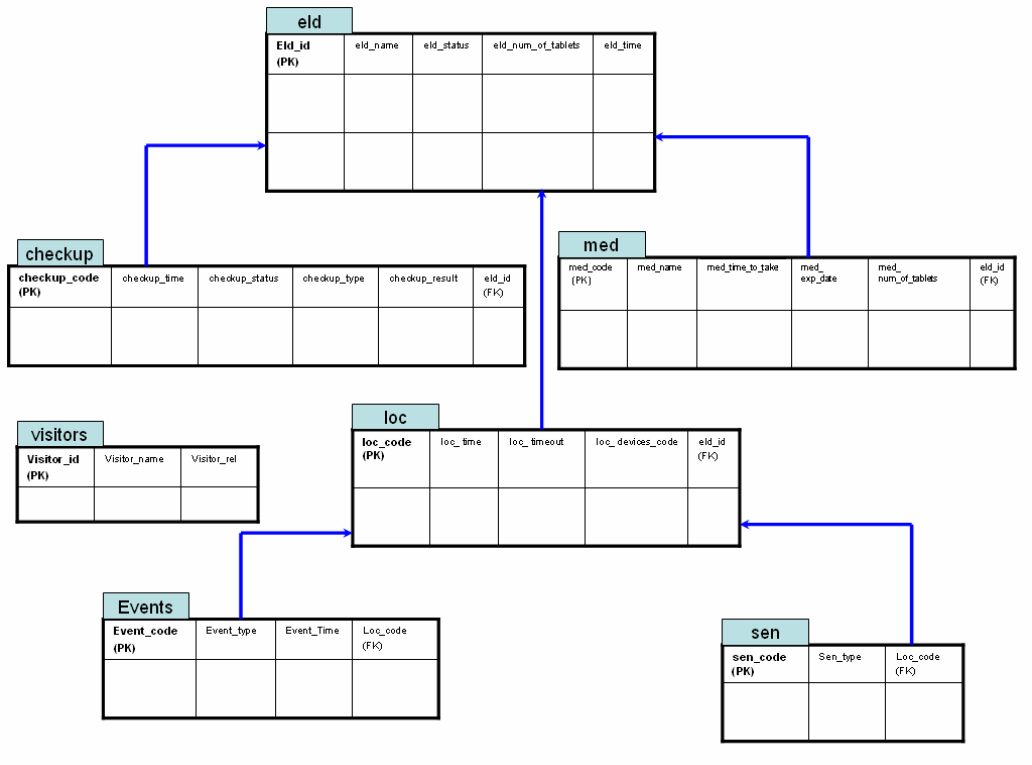


Figure 5.13

Each sensor was numbered so that it can be known and have an entry in the database. These numbers were carefully selected according to location and the type of the sensor. These numbers are as follows:

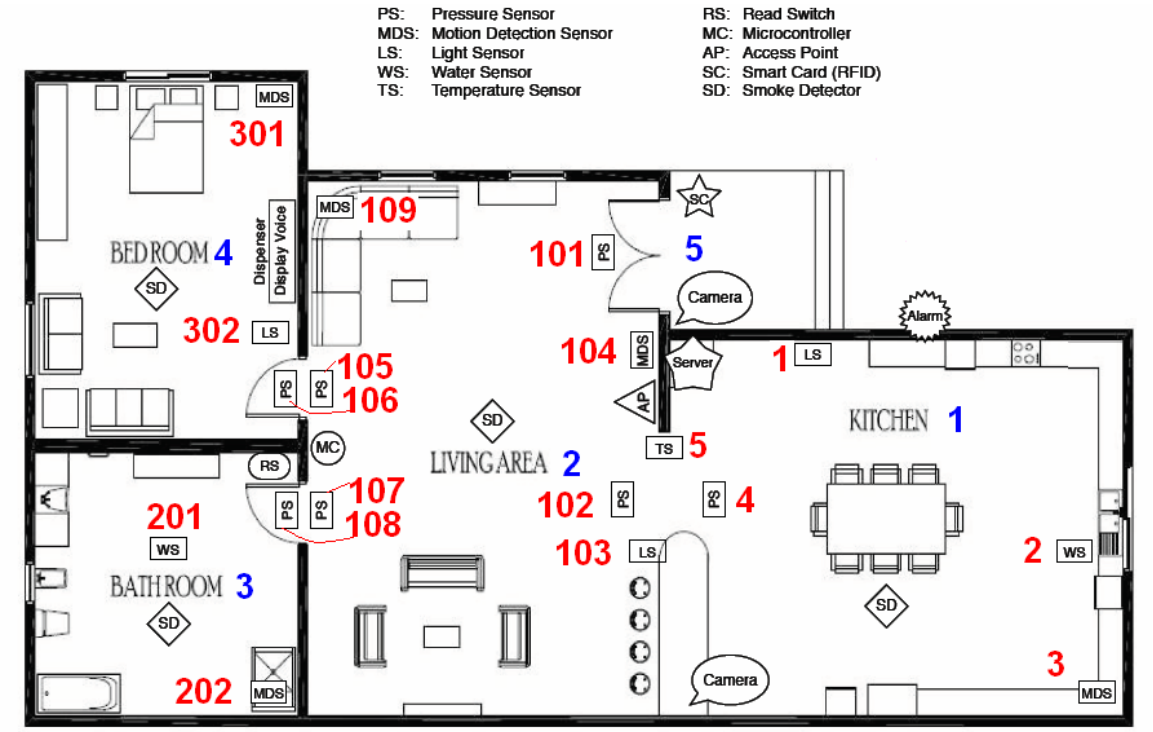


Figure 5.14

2- Client/Server Application:

1- After establishing a successful client/server session with a micro controller-like end system (telnet) to the application, the next step was to forward the information to the data base with the date and time stamp for each record. Details of this application are as the following:

A. Searching for means to establish a connection to a data base:

After studying many references and testing some of those, using the ODBC Command connection commands was found to be convenient for the project needs.

B. Building a data base of our own:

There had to be a data base of this application for testing the forward of the data to it. After that was successfully accomplished, the local database must forward all the records to the database of the system to store the records over there.

C. Integration Process:

The process of integration was done through a simulation of the sensors signals coming from the microcontroller. As mentioned, each sensor has an ID. This ID is used from the microcontroller side too to send signals to the application. Then

these signals are translated into meaningful information so they can be displayed on the application.

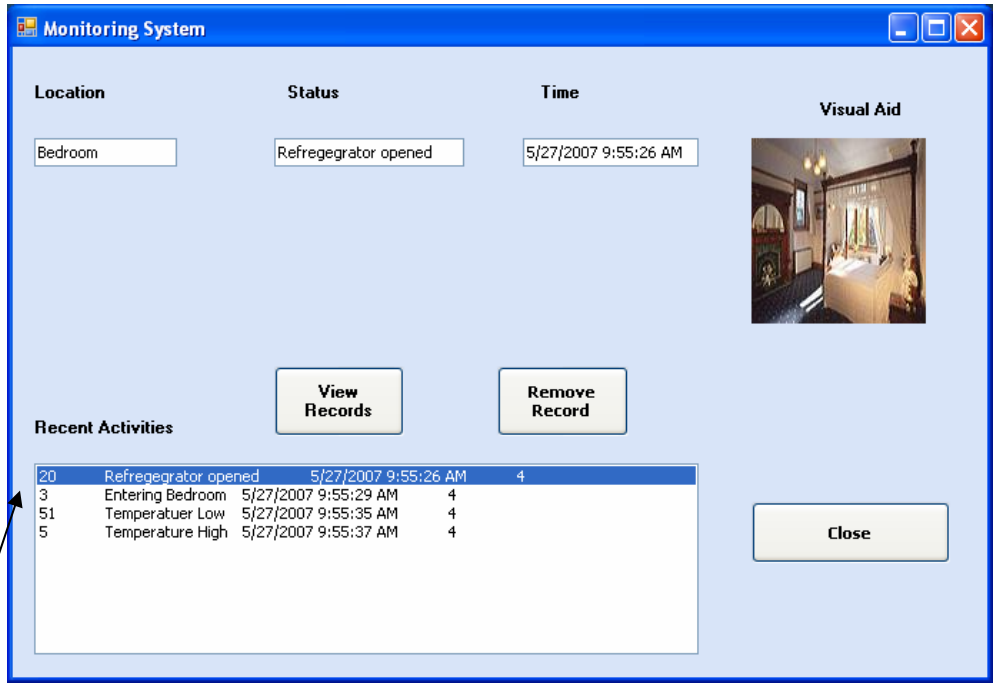


Figure 5.15

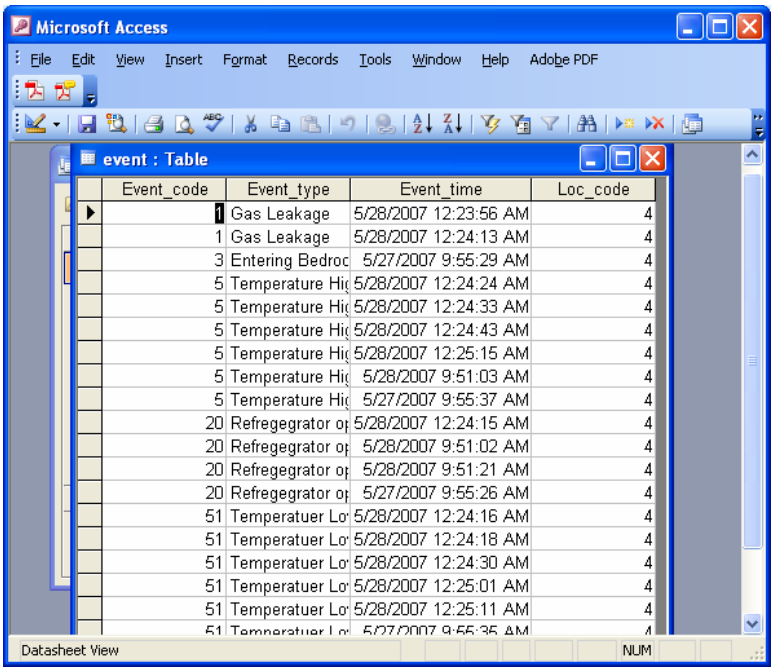


Figure 5.16

3. Additional Features:

A. Built-in data base:

Te local database functions exactly as the data base that was developed for the system. That been said, it requires it to save all the incoming interpreted signals from the microcontrollers and the ability to call it at any time. **Several** reasons to have another data base in this project and they are the following:

1. Mainly as a backup data base in case of failure of the original.
2. Since the application is installed on the PC located at the home of the elderly, the user (relative or visitor) of this PC can view the data base without the need of accessing the web.

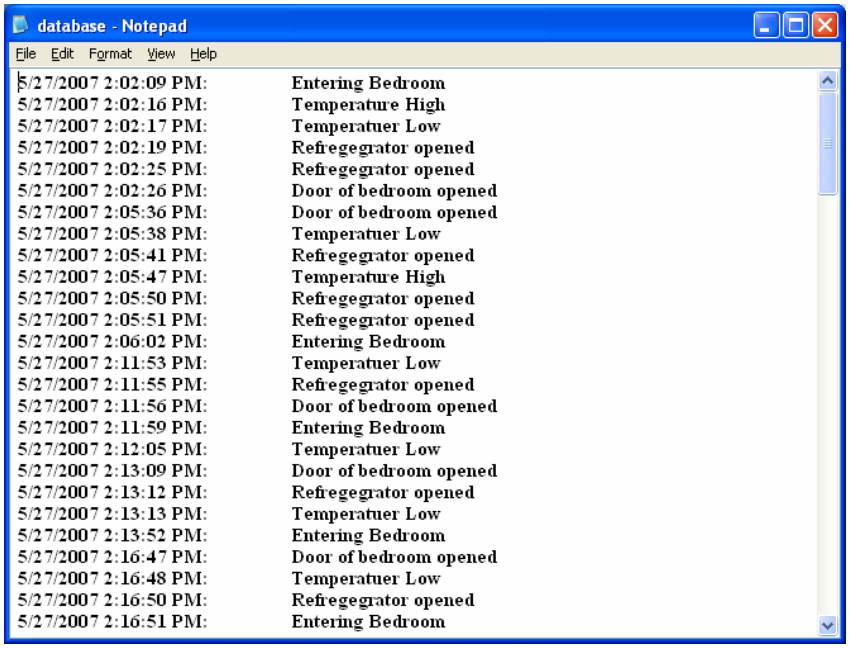


Figure 5.17

B. Friendly-user interface:

That provides a visual aid that can show the location of the elderly, text fields that displays that time, status & location. In addition, a nice feature was used in the user interface, which is a queue was implemented, that once a signal is received it will be displayed there and any consecutive signals will be received and displayed there as well. The user can choose any of those by clicking on to view additional details about it in the text fields however the signals is already forwarded to the data base and does not require any user interaction like clicking or anything of that sort (Real time with the help of multithreading concept).

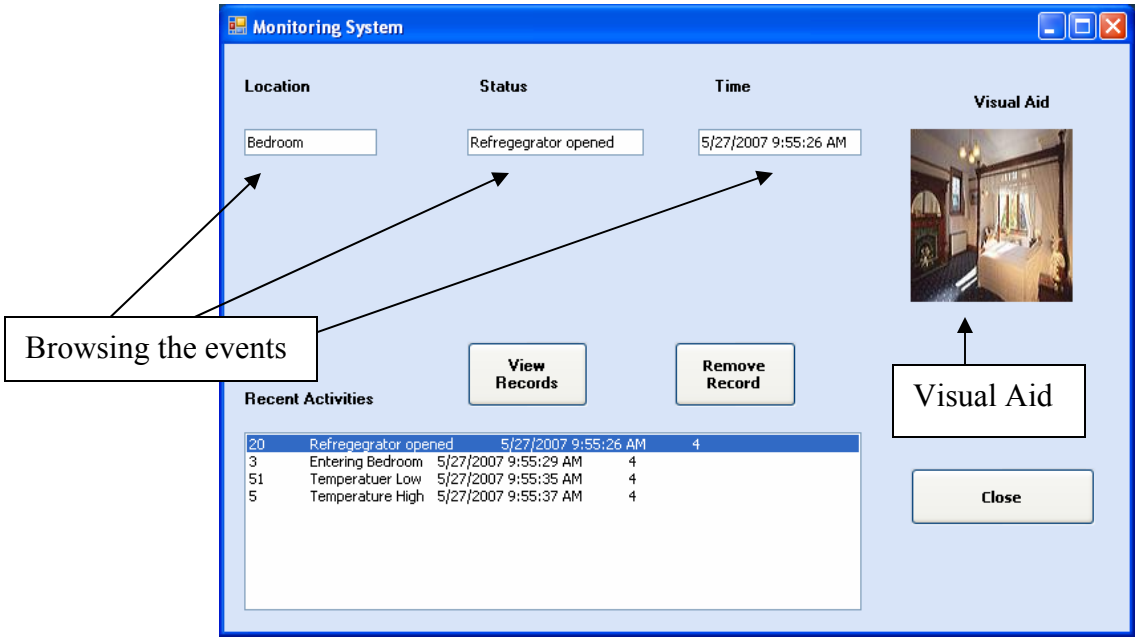


Figure 5.18

C. Error Handing and remove Button:

To remove any record from the list box of the coming events a remove button was added to this job and when there is no record is received yet an error message will be displayed.

D. Warning message with sound Alert in case of emergency:

When the application receives a signal from high temperature signal, it will show a warning message with sound alert for the elderly to protect him/her from any injuries caused by fire. Also, it will show from which sensor this alert comes.

Figure 5.19

3- Web Application:

The web application main idea is to connect the elderly to the outside world by connecting to the database and retrieve the entire information (figure). It was programmed with Visual Basic .Net. It consists of four main sections: patient, doctor, visitor and Map.

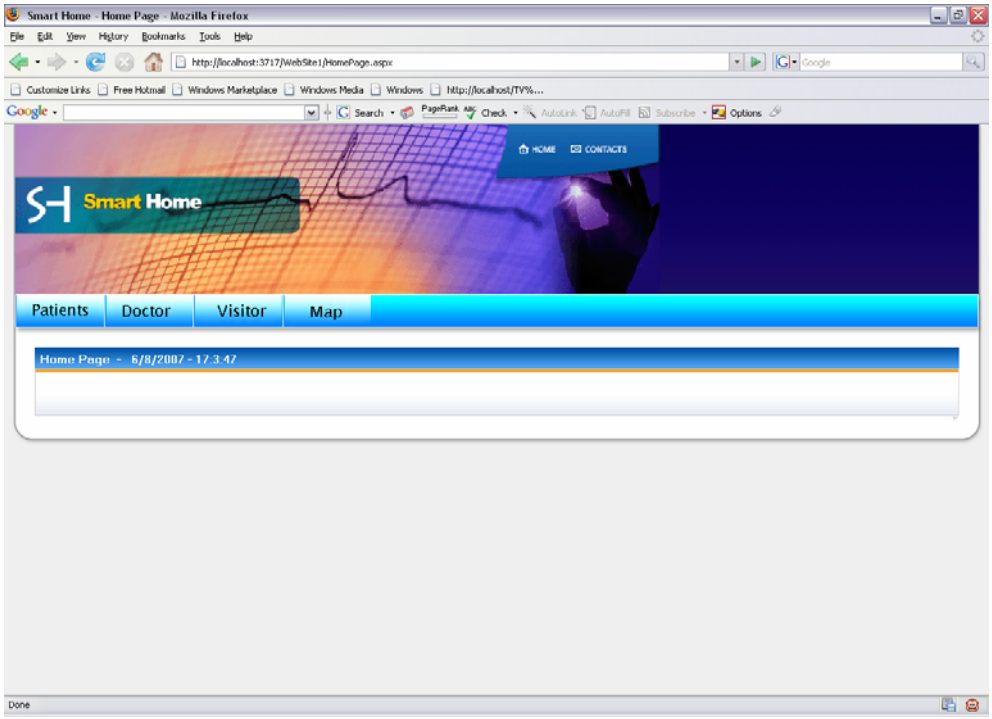


Figure 5.20

1- Patient:

The patient section consists of three parts adding, updating and deleting a patient figure 1.1.

- Adding patient figure 1.2 after entering all the information the web application will connect to the database and insert all the information if some there is an error with some of the information the data will not enter the database.

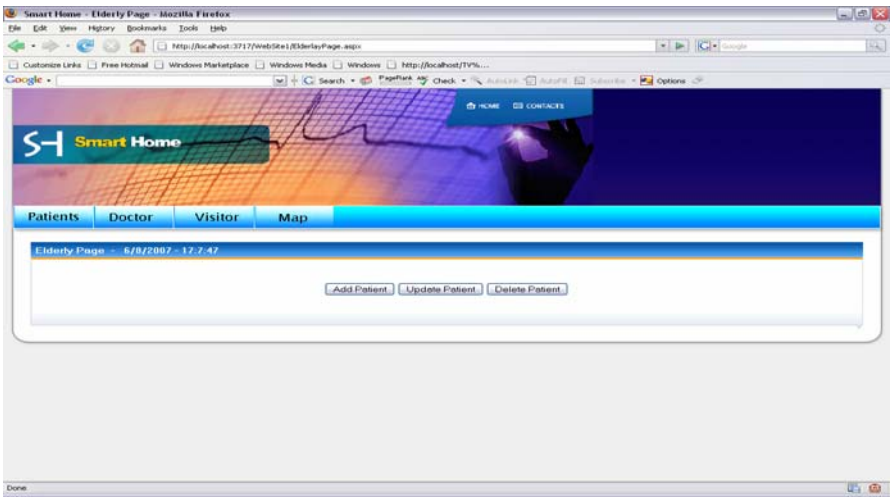


Figure 5.20

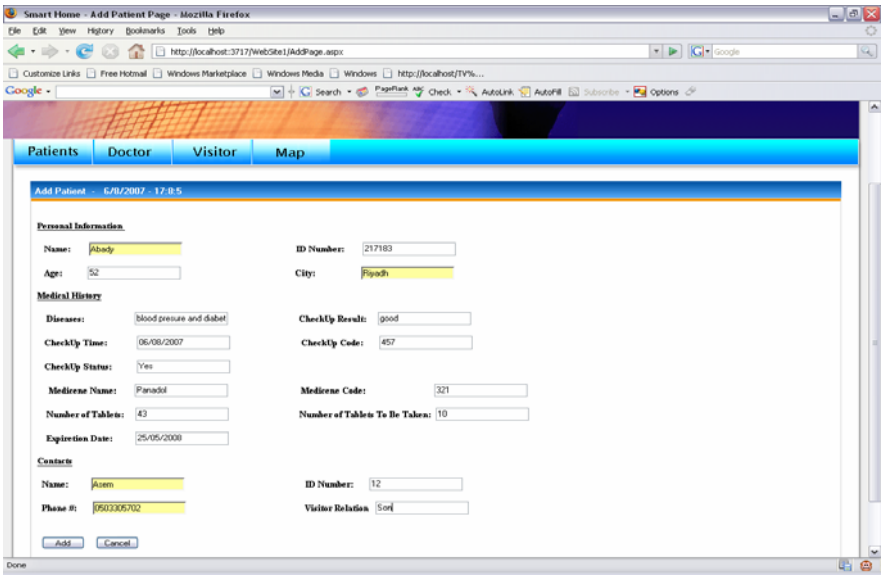


Figure 5.21

- Updating patient first the doctor should choose patient name by clicking the input load the web application will connect to the database and retrieve the entire patient name in the database in the “eld” table and insert them in a dropdown list figure 1.3. Then the doctor will choose the name and update the patient information by entering a new data. Finally when the doctor click the update button the web application will connect to the database and update the patient information.

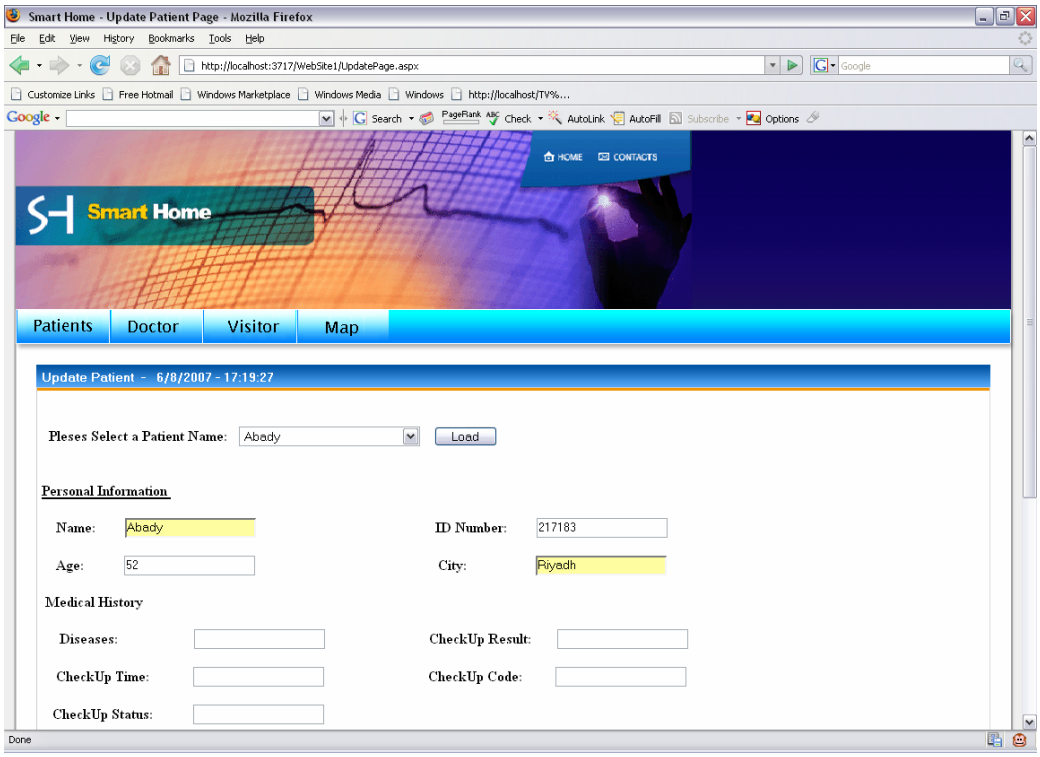


Figure 5.22

- Deleting patient first the doctor should choose patient name by clicking the input load the web application will connect to the database and retrieve the entire patient name in the database in the “eld” table and insert them in a dropdown list figure 1.4. Then the doctor will choose a name and click the delete button then the web application will connect to the database and delete all the information in the data base with the chosen name.

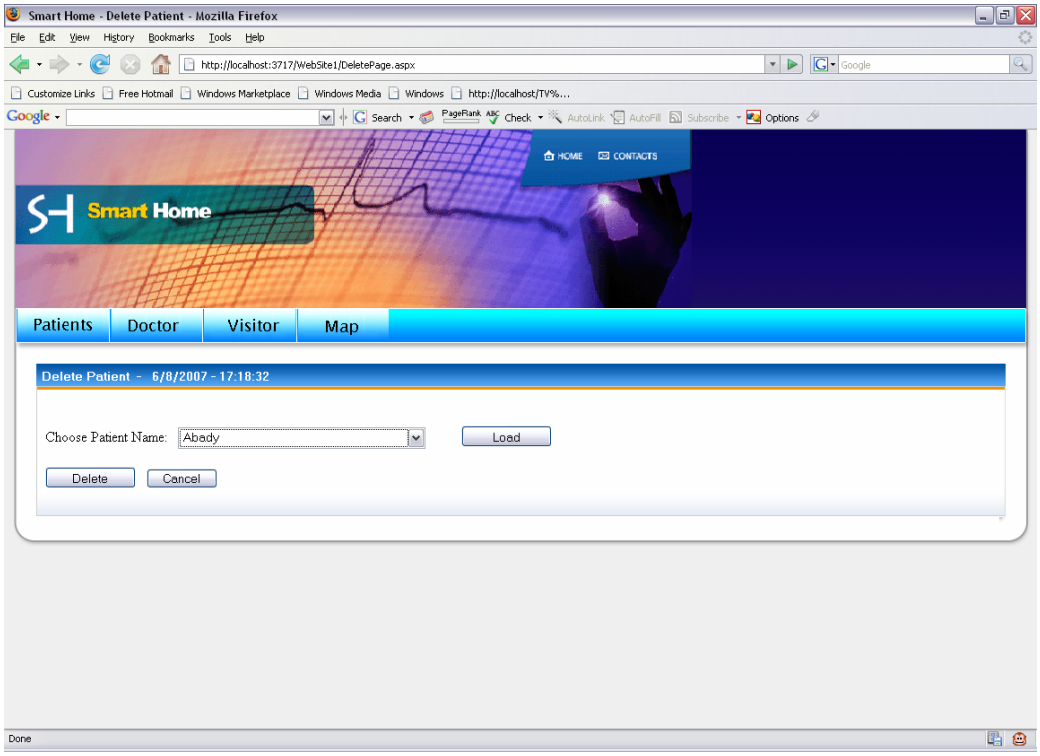


Figure 5.23

2- Doctor:

In this section the doctor can see a full report about the elderly the doctor will only choose patient name and the web application will connect to the database and a full report will be generated about the patient figure 2.1.

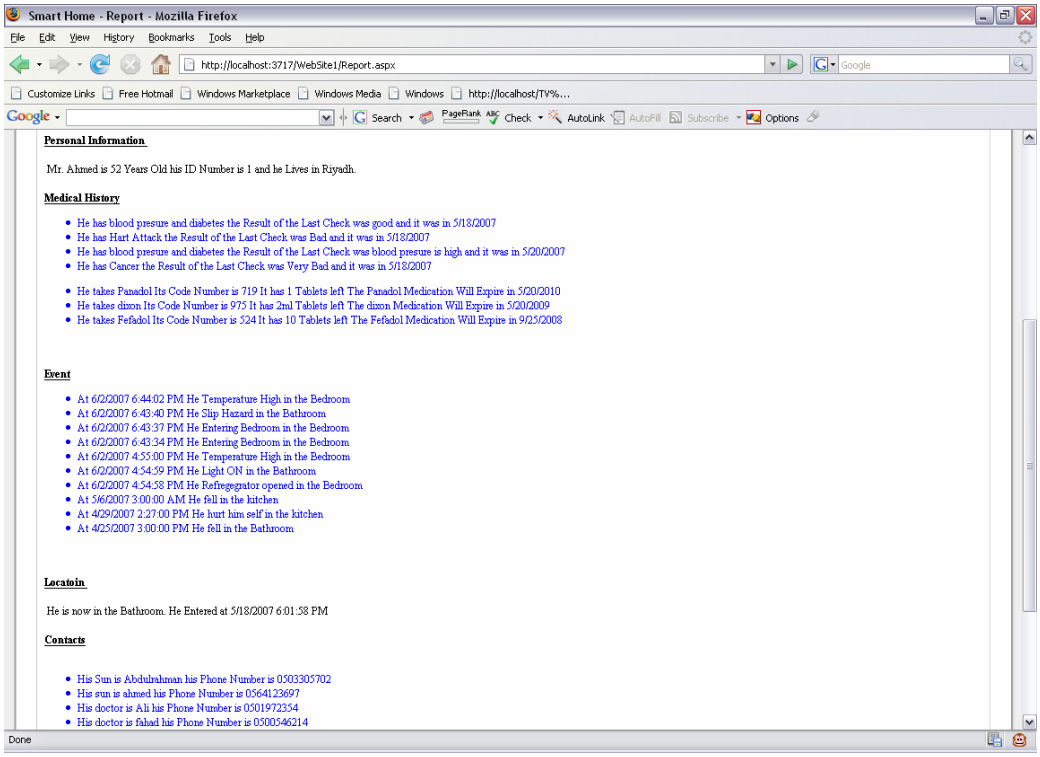


Figure 5.24

3- Visitor:

This section only the visitor can view it the relative will only choose name and the web application will connect to the database and a small report will be generated about their relative figure 3.1.

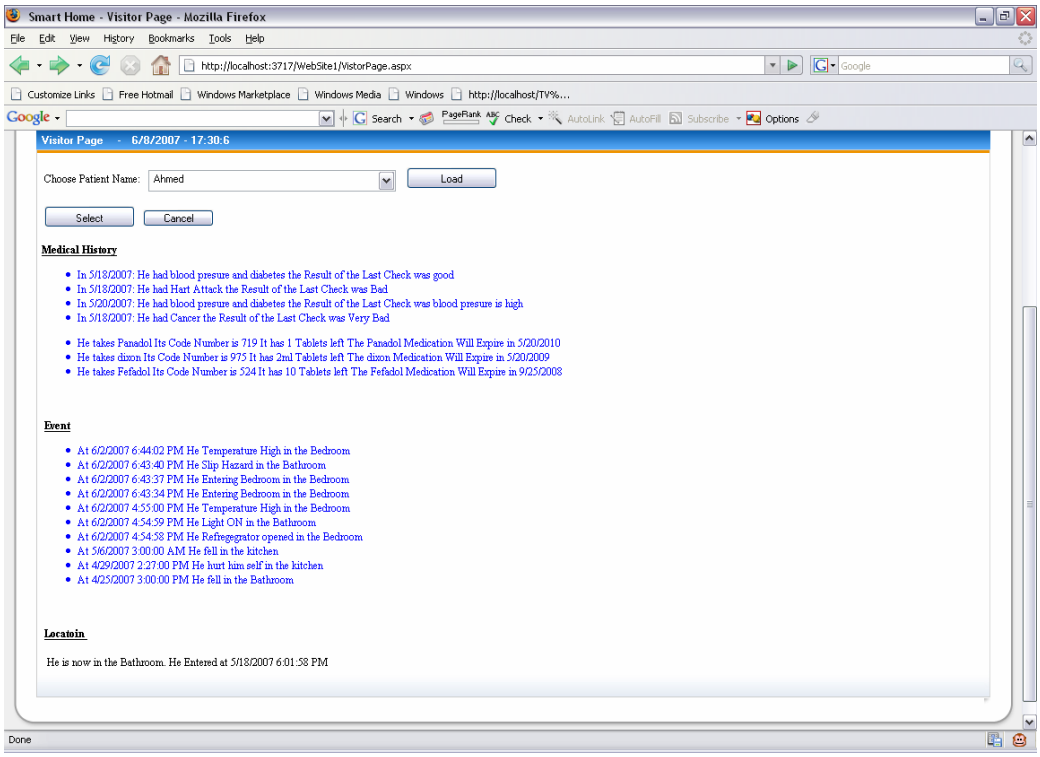


Figure 5.25

4- Map:

In this section the doctor and the visitor can know where the elderly is Figure 4.1 the web application will connect to the data base and check where is the elderly now. This part also contains a direct link to the camera in the main gate to see the entire event around the main gate.

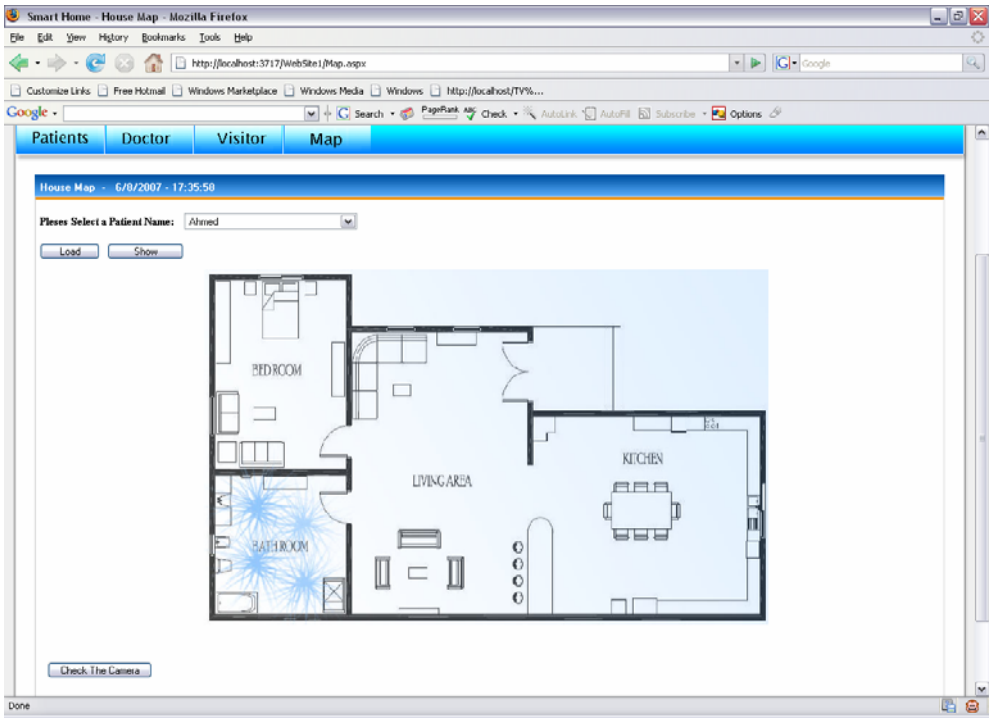


Figure 5.26

4- Communication:

Communication part in this project is concerned with smart card application and wireless connection to the wheel chair. The client application is a program of the smart card that is running on the PC. The PC will be connected to the smart card reader. This part of the project will allow or deny the access of the visitors who want to visit the elderly on his house. So, it needs to access the database of the smart home and communicate with the other applications that use the database. The connection to the wheelchair wirelessly is provided by using wireless transceivers.

a- Smart Card Client Application:

This part is running on the PC which is connected to the smart card reader through the USP port. The following steps will be performed after the smart card is connected to the smart card reader:

- The application will read the data from the smart card. These data will be received in the APDU format.
- Then, it will get the name, id and relation of the visitor.
- These data will be sent to the database as a query to check the availability of the fields (name, id and relation) on the database.
- Then the response will be received from the database to allow or deny the access.

The design is as follows:

ISO-7816

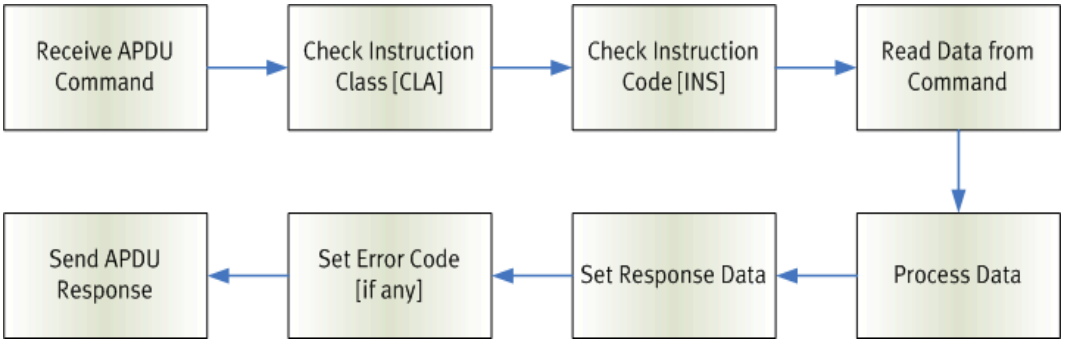


Figure 5.27

- APDU Commands that are sent to the smart card application to perform a specific task

Types:

- Command APDU:
  - ☒ From reader to smart card.
  - ☒ Format:

  - CLA - Class of Instruction: indicates structure and format for a category of commands. [1 byte]
  - INS – Instruction Code: specifies the instruction for the command. [1 byte]
  - P1 & P2 – Instruction Parameters: used as parameters for the instruction. [1 byte each]

- Lc – Length of Command Data: indicated the size (in bytes) of the data field of the command. [1 byte]
- Data Field: optional field to send data with the command. [variable size]
- Le – Length of Expected Response: Maximum length of the data expected in the response. [1 byte]

- Response APDU:
  - ☒ From smart card to reader.
  - ☒ Format:

- Data Field: optional field to send data with the response. [variable size]
- SW1 & SW2 – Status Words: indicated the state of command processing and response. [1 byte each]

- There are 4 ISO cases for any command, I used these tow ISO in My Java code :
  - Case 2: No Data In, Data out.
    - The APDU command has no data, but the response contains data.
  - Case 3: Data In, No Data Out.
    - The APDU command carries some data, but the response has no data.

The main idea for smart card client application is to design GUI interface to run it in the control computer. When a visitor come and uses his smart card, the client application first will display his field, which stored in the java code, then connects to database to check if the person is authorized or not, then it will take an action (either send a signal to microcontroller to open the door if he authorized, or the door will not open). The tasks in this application are:

1. **Initiate smart card connection and exchanging data:**

To be able to do this task, the SmartDeckTerminalTypeLib was downloaded and add it as a reference, the main libraries in this application are:

```
using System.Net.Sockets;     \\ use to communicate with microCntrl
using System.Data.SqlClient;  \\ use to communicate with database
using SmartDeckTerminalTypeLib; \\ use to communicate with smart card
reader
using System.Data.SqlTypes;   \\ use to send query to database
```

Two strings are defined to make the application capable to do both contact and contact-less with minor modification:

```
 String contactLess = "SCM Microsystems Inc. SDI010 Contactless Reader
0";
 String contact = "SCM Microsystems Inc. SDI010 Smart Card Reader 0";
```

Then an object of smart deck terminal type is initiated and define application's name:

```
String applicationName = "SMART";
SmartDeckTerminalComponentClass smartCardObject = new
SmartDeckTerminalComponentClass();
```

The last step before initiating the connection is to define the Application Protocol Data Unit (APDU), which consists of six fields:

- Class of Instruction (CLA): indicates structure and format for a category of commands.
- Instruction Code (INS): specifies the instruction for the command.
- P1 & P2 – Instruction Parameters: used as parameters for the instruction.
- Length of Command Data (LC): indicated the size (in bytes) of the data field of the command.
- Data Field: optional field to send data with the command. [variable size]
- Length of Expected Response (LE): Maximum length of the data expected in the response.

The CLA and INS must be identical to the code which specified in the java code in smart card java code. The implementation of APDU code is done as the following:

```
        const String commandGetID = "70 31 00 00 20";
        const String commandGetName = "70 32 00 00 20";
        const String commandGetRel = "70 33 00 00 20";
```

After initiating the connection we connect and exchange the data to store and display the field, two methods were written: one to get the relation name and the other method is to get the id:

```
            try
            {   //connection
                disconnect();
                smartCardObject.connectpcsc(contact);
                smartCardObject.selectbyname(applicationName);
                smartCardObject.setcommand(commandGetID);

                smartCardObject.exchange(); // communicate
                                Byte[] bID = new Byte[20];
                for (short j = 0; j < 20; j++)
                bID[j] = smartCardObject.getresponsebyte(j); //store id
                label6.Text =
System.Text.ASCIIEncoding.UTF8.GetString(bID); //display the id
in GUI

                smartCardObject.setcommand(commandGetRel);
                smartCardObject.exchange(); // communicate
                                Byte[] bRel = new Byte[20];
                for (short j = 0; j < 20; j++)        //store relation
                    bRel[j] = smartCardObject.getresponsebyte(j);

                label7.Text =
System.Text.ASCIIEncoding.UTF8.GetString(bRel); //display relation

                disconnect();
                            }



            catch
            {
                err.Text = "Error Connecting to SmartCard";
            }
```

1. Socket programming:

In this application, socket programming is needed to communicate with the microcontroller in case the smart card is authorized and will send an open signal to the microcontroller to open the door. TCP communication is used for more reliability. However, sending data depends completely in the database query dedicated for the application. TCP listener class is used to initiate the connection and a stream writer to send through the connection is defined:

```
TcpClient client;
StreamWriter writer;
```

The main code for initiating the connection (using the microcontroller IP and port number) is:

```
client = new TcpClient("10.13.49.31", 399);
stream = client.GetStream();
reader = new StreamReader(stream);
writer = new StreamWriter(stream);
```

Finally, send the data (after the database authorized the user):

```
writer.WriteLine("Open");
writer.Flush();
```

2. Design clock, last access time and the administrator name:

Additional features in GUI interface were implemented; this includes clock, the last access time and displaying the administrator name.

-Clock: to be able to do the clock, a thread is used to take a time stamp every on second and display it in the GUI. The code is displaying as follow:

```
public void refrishTime()
{


    while (true)
    {
        Thread.Sleep(1000);

        DateTime now = DateTime.Now;
        DateTime now1 = DateTime.Now;
        try
        {
            label15.Text = now1.ToString();
        }
        catch
        {
        }


    }
```

- Last access time: this is achieved by storing the timestamp right after the administrator login and stores it in text file. The second time the administrator logs in it will read the previous timestamp form text file and displays it. This will include reading and writing form the text file:

```
            TextReader lastacc = new StreamReader("LastTimeAcces.txt");
            label16.Text = lastacc.ReadLine().ToString();
            lastacc.Close();

            TextWriter lastacc1 = new
StreamWriter("LastTimeAcces.txt");
            lastacc1.WriteLine(DateTime.Now);
            lastacc1.Close();
```

- Display the administrator's name: read the user name form the text file that contains username and password, and display it in the GUI.

```
            TextReader usps = new StreamReader("UserPass.txt");
            label14.Text = usps.ReadLine().ToString();
            usps.Close();
```

The GUI is shown in the following figure:

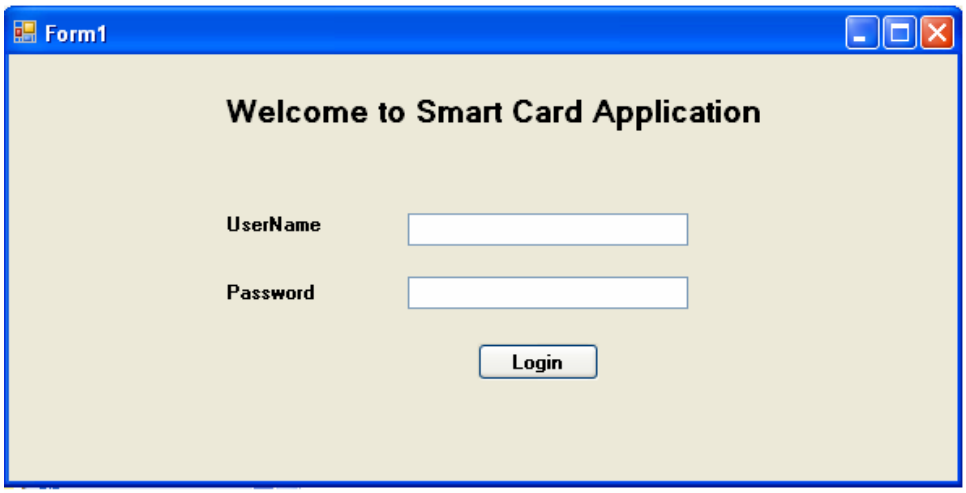


Figure 5.28

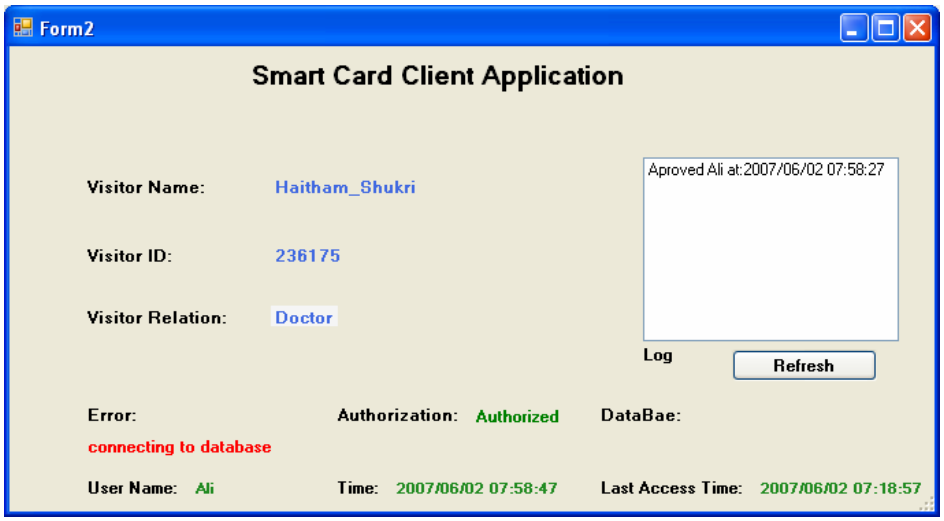


Figure 5.29

b- Wheel Chair Application:

This application is implemented in the laptop in the wheel chair. The objective for this application is to take the data which stored in text file (will be supplied by the serial port) and initiate a connection to the main application and send it. The following steps will be performed:

- The microcontroller will put its data in text file.
- Then, the application will read from the text file.
- After the connect button is pressed, the connection with the application will be established.
- Then, establish a stream object for sending the data.
- After that, it will send the data to the application, which is running on the main PC.
- These data will be sent by opening a socket on the PC which is connected to microcontroller of the wheelchair.
- The data will be sent in ASCII format to the application on the main PC.
- The IP address of the main PC and the port number of the application will be used to communicate with the application running on the main PC.

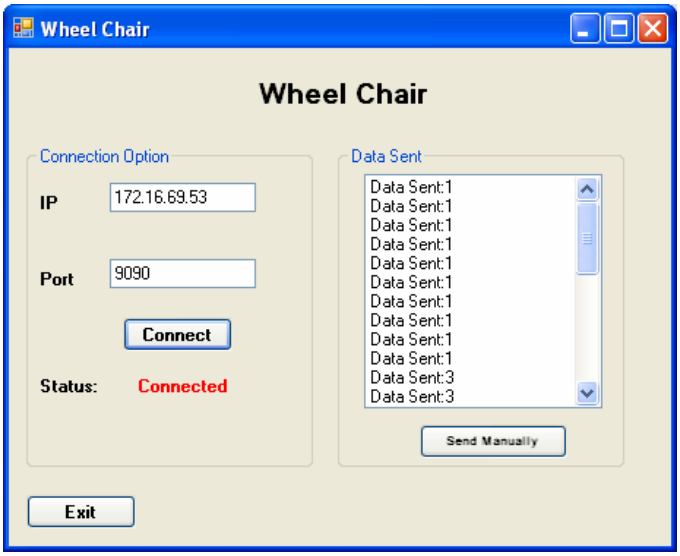


Figure 5.30

C- Hardware:

1- Sensors:

Many sensors are used in this project as mentioned in the design requirements. The design and implementation of these sensors will be discussed throughout this part of the report:

a- Medical Sensors:

Medical sensors were attached in the wheelchair to ensure safety for the elder. They are interfaced with the system using PIC microcontroller.

- CO Design:

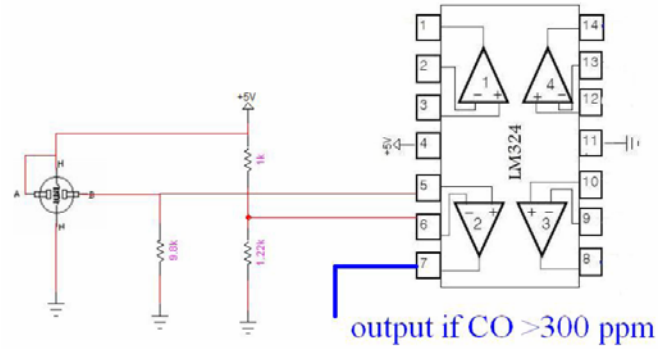


Figure 5.31

- Schmitt trigger:
The presence of noise variations may cause troublesome rapid cycling at the switch threshold. This problem is overcome in the Schmitt trigger circuit by setting a lower threshold for the switching when the voltage is going down. Schmitt trigger has two threshold voltages one positive and the other is negative so, if the signal was above positive threshold the output will be high and if the input signal was below negative threshold the output will be low .

- A-stable Mode and Mono-stable Mode
In the circuit designed the two basic modes of operation were used.
1- Mono-stable Mode
2- A-stable Mode.

In the mono-stable mode is acting as a "one - shot".

T = .693(R C)

T = 3.2 S

In the a-stable mode the 555 will retrigger itself to output a stream of pulses of variable length during the one pulse period in mono-stable mode.

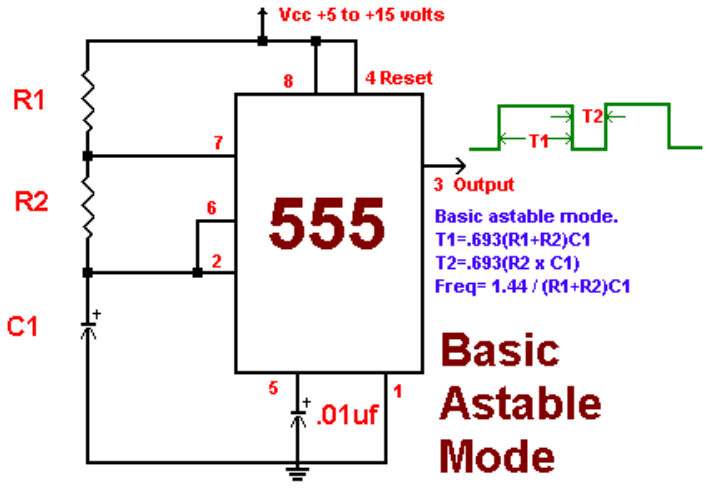


Figure 5.32

To calculate the T1 time (output high) use the following formula T1 = .693(R1 + R2)C1.
To calculate the T2 time (output low) use the formula T2 = .693(R1 x C1).

So,T1=0.9 S,T2=0.2 THE TOTAL =1.1

So, two clock cycles are needed during one clock cycle in the mono-stable mode, one for to prepare the number in the mobile the other clock is for dialing the number (for more information see the datasheet for 7414, 74121, 555). The final design of the medical sensors is as follows:

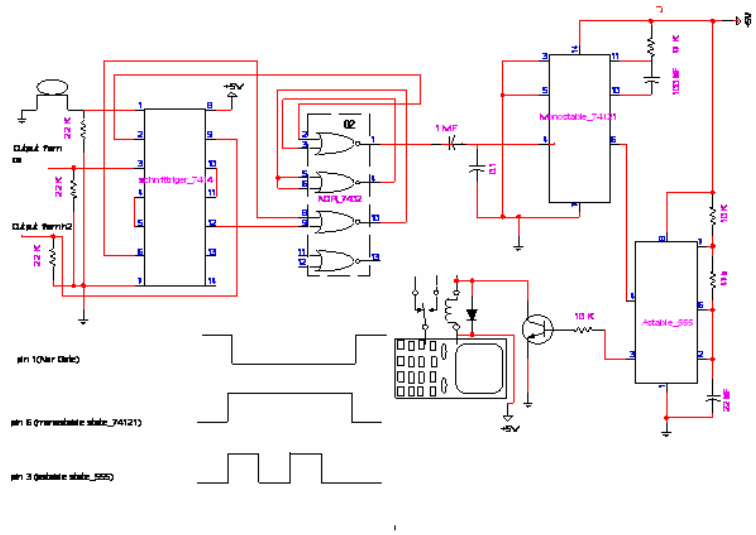


Figure 5.33

- The Blood Pressure:

This devise has its own software that reads records from it and write them to a file. This file is called PBFile.csv and it is stored in C:\Program Files\BPFile\Data\PBFile.csv.
A java code was written that reads records from the .csv file and writes them to the data base. This code can automatically and silently run after any period of time using windows scheduled tasks (See Appendix).

This is the PBFile.csv:

	A	B	C	D
1	PatientID,			
2	NO.,DATE,TIME,SYS,DIA,PLS			
3	1,2006/11/21,15:49:14,103,68,87			
4	2,2006/11/21,15:50:18,108,69,75			
5	3,2006/11/21,15:51:26,112,75,79			
6	4,2006/11/21,16:38:26,115,78,66			
7				

Figure 5.34

This is the data base before running the code:

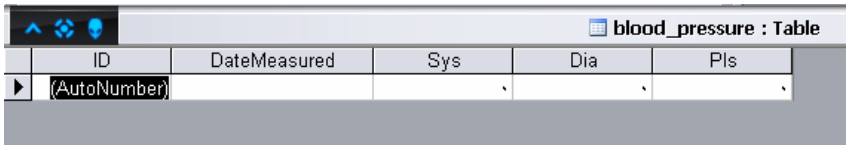


Figure 5.35

This is the data base after running the code:

blood_pressure : Table

	ID	DateMeasured	Sys	Dia	Pls
▶	٣٣١	٢١/١١/٢٠٠٦	١٠٣	٦٨	٨٧
	٣٣٢	٢١/١١/٢٠٠٦	١٠٨	٦٩	٧٥
	٣٣٣	٢١/١١/٢٠٠٦	١١٢	٧٥	٧٩
	٣٣٤	٢١/١١/٢٠٠٦	١١٥	٧٨	٦٦
*	(AutoNumber)		٠	٠	٠

Figure 5.36

b- Location Detection:

This application is about detecting the location of the elderly in his house. To locate a person in the house, two systems for redundancy are used: 1- Pressure Pad sensors. 2- Motion detection sensor. In every door there will two pressure pad sensors on the floor, one inside the room and the other is outside. If the two sensors "motion- pressure" in the same room detect the person, then this is considered that he/she entered that room.

Design layout:

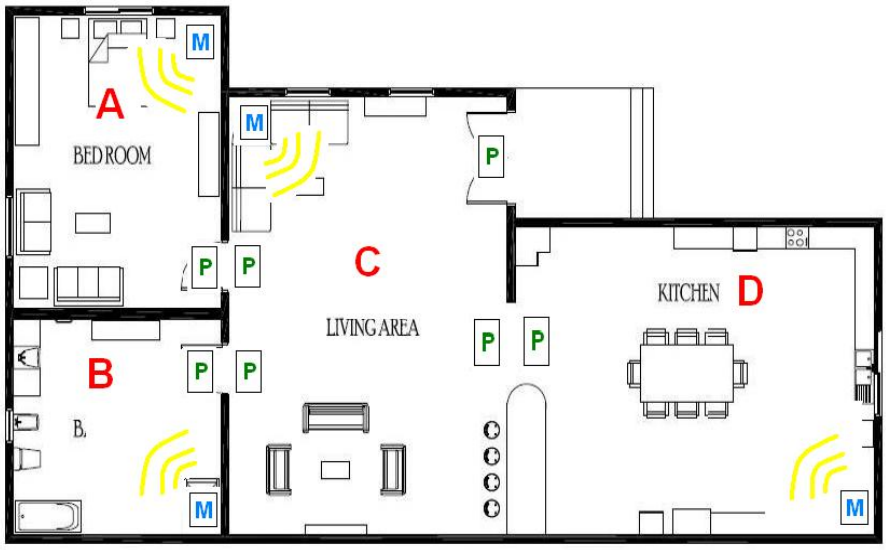


Figure 5.37

A= Bedroom
B= Bathroom
C= living Room
D= Kitchen
P= Pressure Pad Sensor
M= Motion Sensor
So if I say
B_P = Pressure pad sensor in the bathroom
C_M = motion Sensor in the living room

	A_P	B_P	C_P	D_P	A_M	B_M	C_M	D_M
Entering Bedroom	1	0	0	0	1	0	0	0
Moving in the bedroom “	0	0	0	0	1	0	0	0
Entering Bathroom	0	1	0	0	0	1	0	0
Moving in the bathroom “shower”	0	0	0	0	0	1	0	0
Entering living room	0	0	1	0	0	0	1	0
Moving in the living room “watching TV”	0	0	0	0	0	0	1	0
Entering the kitchen	0	0	0	1	0	0	0	1
Moving in the kitchen “cooking”	0	0	0	0	0	0	0	1
Ignore Rest								

Main Compounents:

1- Presure Pads:

Floor pressure mats or pads are designed to detect a person treading on them. Their primary use is in the security industry. They offer a low cost covert method of detecting a person and will provide years of service. The pressure mat / pad should be mounted on a flat and even floor then covered with a conventional floor covering such as carpet. Many other applications have been found for these floor pressure mats / pads such as, sensors for interactive toys, sensors for interactive multimedia systems, i.e. when a person treads on a floor mat a multimedia presentation is triggered therefore targeting the presentation accurately.

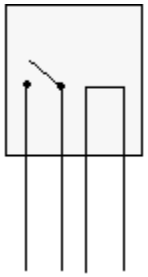


Figure 5.38

Pressure Mat / Pads are available with either four wires for the security industry or two wired normally open for non-security applications.

Key benefits:

1- low cost
2- Reliable

2- **Motion Detector:**

There are many ways one can detect nearby human or object motion

IR motion detection

Infrared radiation exists in the electromagnetic spectrum at a wavelength that is longer than visible light. It cannot be seen but it can be detected. Objects that generate heat also generate infrared radiation and those objects include animals and the human body whose radiation is strongest at a wavelength of 9.4um. Infrared in this range will not pass through many types of material that pass visible light such as ordinary window glass and plastic. However it will pass through, with some attenuation, material that is opaque to visible light such as germanium and silicon. An unprocessed silicon wafer makes a good IR window in a weatherproof enclosure for outdoor use. It also provides additional filtering for light in the visible range.

Pyroelectric Sensors**:** The pyroelectric sensor is made of a crystalline material that generates a surface electric charge when exposed to heat in the form of infrared radiation. When the amount of radiation striking the crystal changes, the amount of charge also changes and can then be measured with a sensitive FET device built into the sensor. The sensor elements are sensitive to radiation over a wide range so a filter window is added to the TO5 package to limit detectable radiation to the 8 to 14mm range which is most sensitive to human body radiation.

Typically, the FET source terminal pin 2 connects through a pulldown resistor of about 100 K to ground and feeds into a two stage amplifier having signal conditioning circuits. The amplifier is typically bandwidth limited to below 10Hz to reject high frequency noise and is followed by a window comparator that responds to both the positive and negative transitions of the sensor output signal. A well filtered power source of from 3 to 15 volts should be connected to the FET drain terminal pin 1.

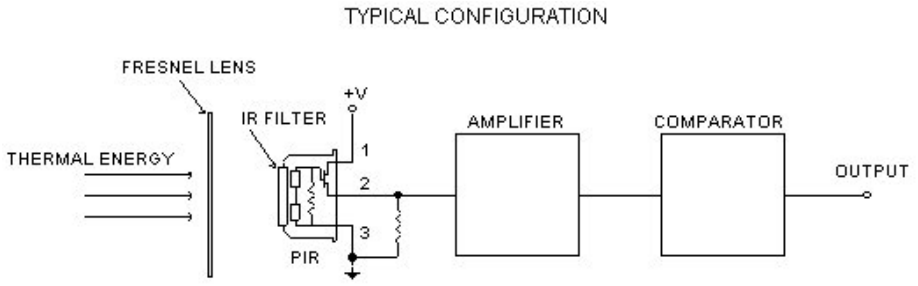


Figure 5.39

The PIR325 sensor has two sensing elements connected in a voltage bucking configuration. This arrangement cancels signals caused by vibration, temperature changes and sunlight. A body passing in front of the sensor will activate first one and then the other element whereas other sources will affect both elements simultaneously and be cancelled. The radiation source must pass across the sensor in a horizontal direction when

sensor pins 1 and 2 are on a horizontal plane so that the elements are sequentially exposed to the IR source. A focusing device is usually used in front of the sensor

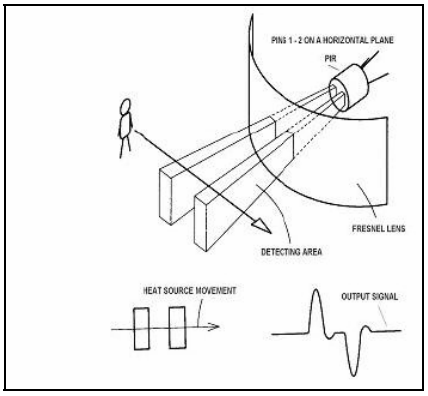


Figure 5.40

The figure below shows the PIR325 electrical specifications and layout in its TO5 package. Note the wide viewing angle without an external lens.

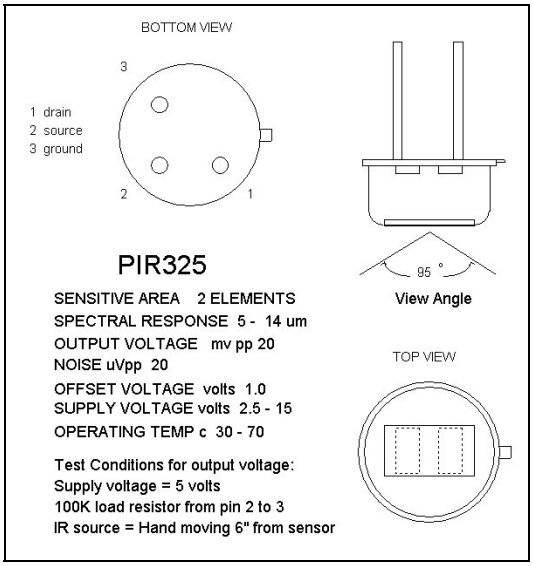


Figure 5.41

This is a typical application circuit that drives a relay. R10 and C6 adjust the amount of time that RY1 remains energized after motion is detected

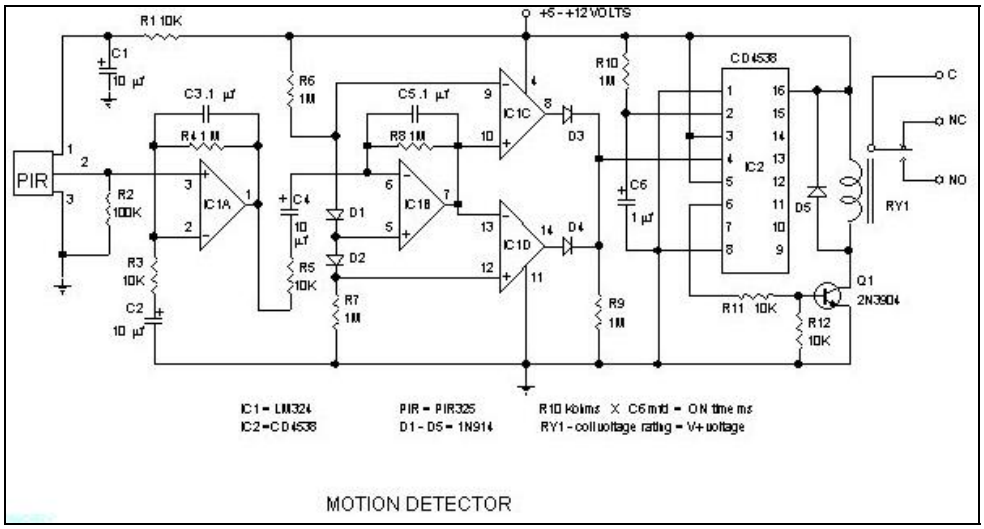


Figure 5.42

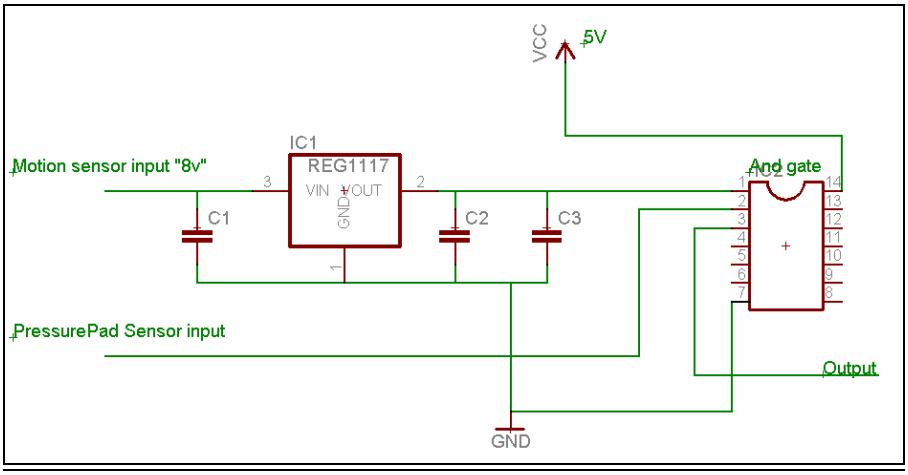


Figure 5.43 (Final output to the microcontroller)

c- Gas & Smoke Sensors:

Design and implement three environmental sensors: Carbon Monoxide (CO) sensor, Temperature sensor, and Liquefied Petroleum Gas (LPG) and Propane sensor.

- Location of the Sensors in the House

As stated in the project plan, the temperature and the CO sensors will be located in three different places: the bedroom, the living area, and the kitchen. Moreover, a third sensor is added that should be located in the kitchen near the oven in order to detect the LPG gas leak which will be its only location. As shown in figure 01 below

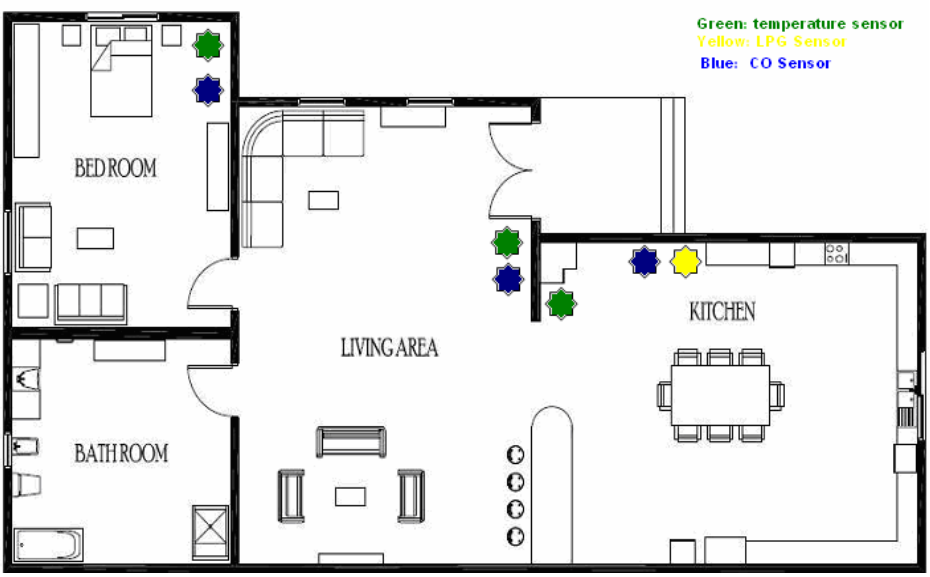


Figure 5.44

1- Carbon Monoxide Sensor

CO is a colorless, odorless, toxic gas. It is produced by the incomplete combustion of solid, liquid or gaseous fuels. Appliances fueled with gas, oil, kerosene, wood or coal may produce CO. If such appliances are not installed, maintained, and used properly, CO may accumulate to dangerous levels. All fuel burning appliances have the potential to produce CO in varying concentrations.

- Circuit Design

After getting the sensor, the analysis method to the circuit according to the datasheet is as follows:

RL = 9.8 K Ω; from the DS

$$\frac{Rs}{Ro} = RL * \frac{(VC - VRL)}{VRL}$$

In order to get R_0 we have to get it in clean air (with 100 ppm)

Ro = Rs (AT VRL = 1.85V) self testing

$$Ro = 9.8K * \frac{(5 - 1.85)}{1.85}$$

Ro =16.686 K Ω

But we need to calculate the VRL so that it gives us the monoxide concentration

of 300 ppm ➔ $\frac{Rs}{Ro} = 0.48$ from the D.S.

Rs = 0.48 * Ro = 0.48 * (16.686k) = 8 K Ω

$$\frac{Rs}{Ro} = RL * \frac{(VC - VRL)}{VRL}$$

$$8\text{ k} = 9.8K * \frac{(5 - VRL)}{VRL}$$

8 k VRL = 49 k – 9.8 k VRL

17.809k VRL = 49k

VRL = 2.75 volt (at 300 ppm)

So, now we no know the voltage level we want the comparator to take action on.
So, let's calculate resistors R1, R2

Assume R1 = 1k Ω

$$Va = Vc * \frac{R2}{R1 + R2}$$

$$2.75 = 5 * \frac{R2}{1 + R2}$$

R2 = 1.22 k Ω we got it by a variable resistor

As you see in the circuit design.

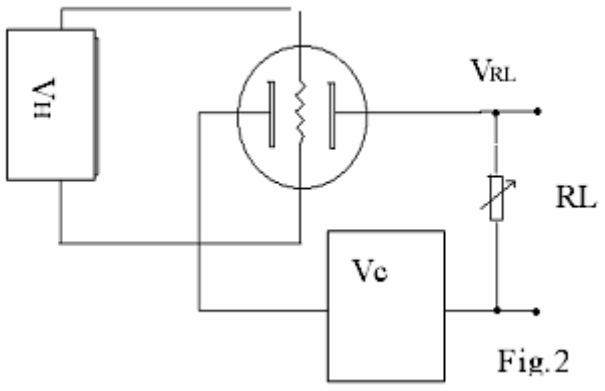


Figure 5.45 (Circuit of the CO Sensor)

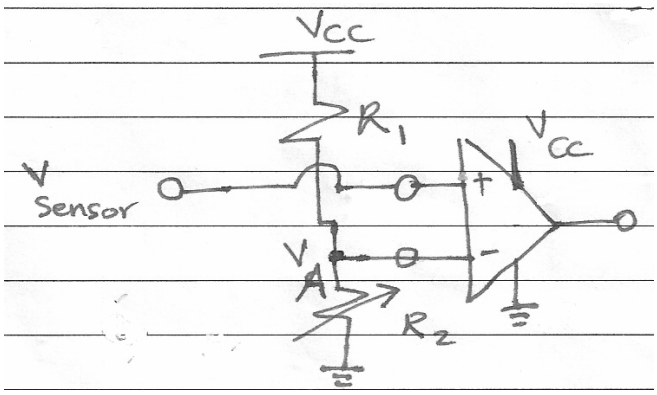


Figure 5.46 (The Comparator used)

2- LPG and Propane Sensor

As part of the environmental sensors the LPG sensor plays an important role if the elder's safety in the kitchen, moreover, it has to be placed near the oven.

- Circuit Design

As shown in the circuit design that the V_{out} will be placed similar to the CO sensor having a comparator instead of the AC to DC converter with RL = 10 K Ω. Moreover, we can have either one of the two terminals connected to the V_{out} (either A or B)

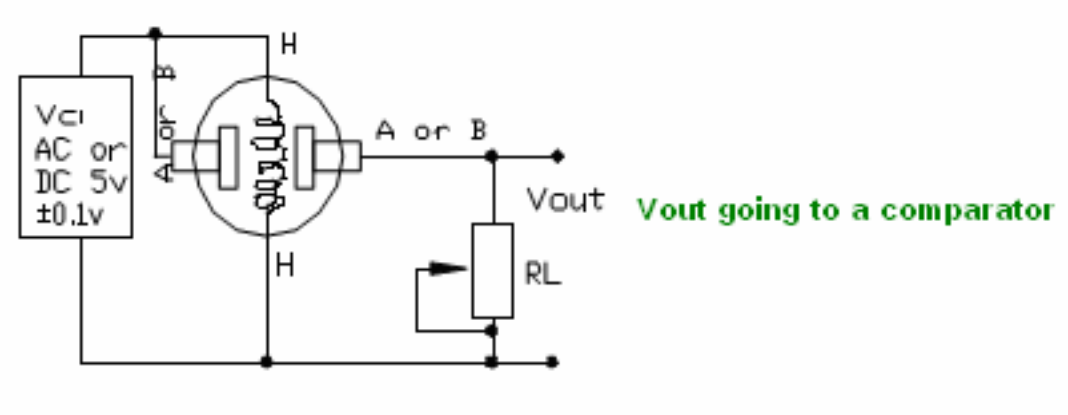


Figure 5.47 (LPG and Propane Sensor)

3- Temperature Sensor

As part of comfortableness that should be provided to the elder the temperature sensor plays an important role in that by adjusting the desired temperature to the elder.

- Circuit Design

After searching the internet for a good sensor, two types were found that exactly meet my requirements. However the first one was not available in the market. Therefore the second one is used, which is the LM35. The LM35 is an accurate and low cost sensor which has a low output current as well.

The following assumptions were made:

- The optimum temp. range is 33^{O} Celsius.
- The temp. should show high if passed 33 to higher degree.
- The temp. should show low if passed 33 to lower degree.
- The system should call for help if passed 39 to higher degree.

After having the output from the sensor, a connection is made to either an AC to DC converter (for more levels), or a comparator for three levels (which is used here). This is shown in the following figure:

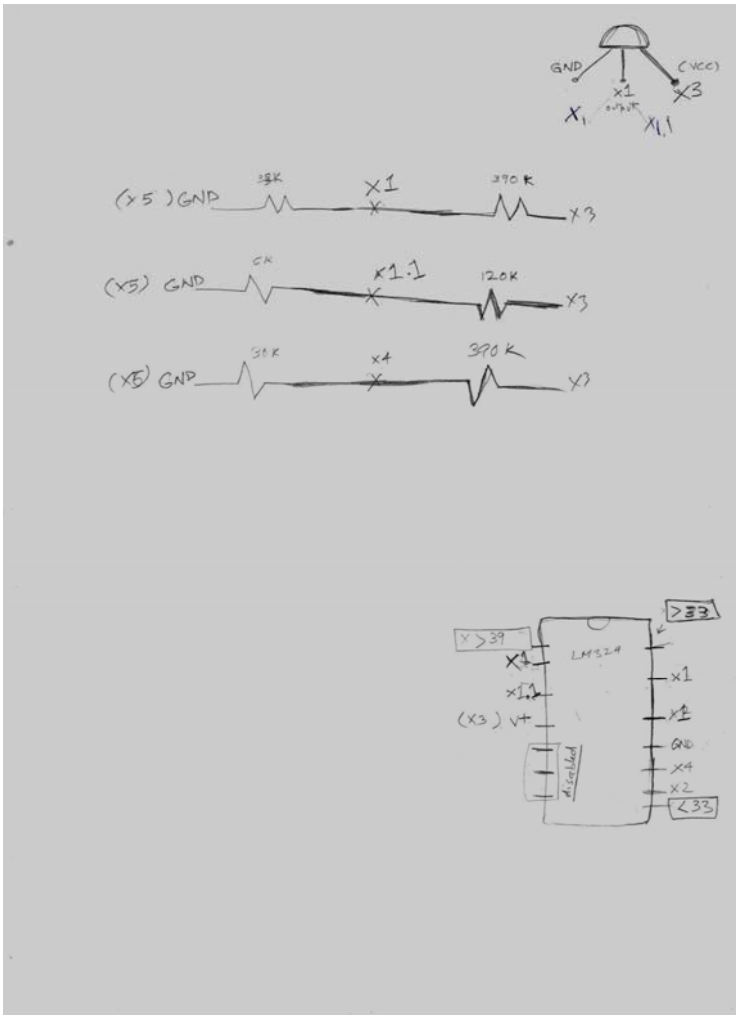


Figure 5.48 (Temperature Sensor Design with the Comparator inputs and outputs)

d- Water Sensor:

Water sensor will detect the availability of the water in the bathroom's floor. The main objective of these tasks is to help the elderly to know if there is water in the bathroom and his/her life can depend on it. Therefore, it assures security of the elder.

- Design:

Water sensor is a hardwired detector for the water in the floor. The main idea of its implementation is to use two probes that are connected to floor in a hidden fashion. These two probes are then connected to the circuit that will analyze it is there water or low level or high.

- Design Descriptions:

The sensor can be made by a simple circuit that takes benefits of water ionization. With the coverage of footstep, the probes can be hidden and detect the water as well. Here you will find the circuit that is implemented to get the result.

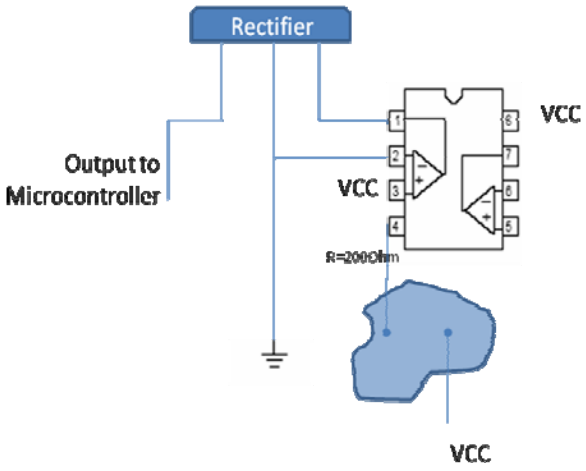


Figure 5.49

- Tradeoffs made:

A tradeoff between the readymade water sensor and this built water sensor is made and this resulted that the sensor will be better in terms of the following:

1- Reliability.
2- Availability.
3- Sensor sizes.

e- Fall Detection Sensor:

Fall detection sensor requires a logical design to be implemented. It can be considered as a small system that has a set of sensors and a microcontroller that detects the sensor's signal and gives the output. The requirements of a fall detection sensor are as follows:

1- PIC Microcontroller:
Receives signals from sensors and gives the output signal to the main system. Three inputs will be used:
   - INT: interrupt for the fall detection.
   - Timer: to count for a period of time while there is no movement.
   - Counter: to count for the movement signals (described below).

2- Movement Sensor:
It is a sensor that detects the movement of an object and it is attached to this object directly. Signals are sent to a receiving object (Microcontroller in this case) while the object is moving. The detection is done by having two spheres inside it that will make a circuit when colliding. Input is called "Move".

3- Laser:
The laser is a redundant input to the circuit. It is used to differentiate between a falling state and a still state.

4- Laser Receiver Sensor:
It is used to detect the presence of the laser. Input is called “Laser”.

Following is a state machine diagram that illustrates the cases of this circuit which will give the desired output in the end:

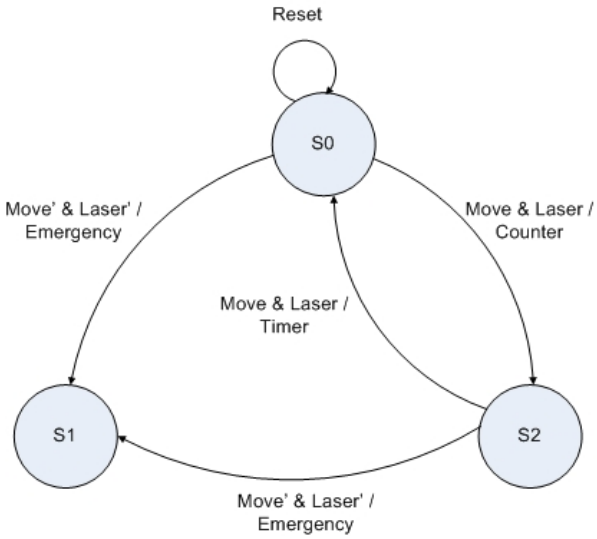


Figure 5.50

S0: Sit/Still
S1: Fall detection
S2: Movement

Truth table can be constructed from this state machine diagram and it is as follows:

Input	**Logic**	**Output**
Move’ & Laser’	00	Fall
Move’ & Laser	01	Sit/Still
X	10	X
Move & Laser	11	Move

From the truth table above, a 2x4 decoder can be used to interface these outputs to the microcontroller. The block diagram for the design is as follows:

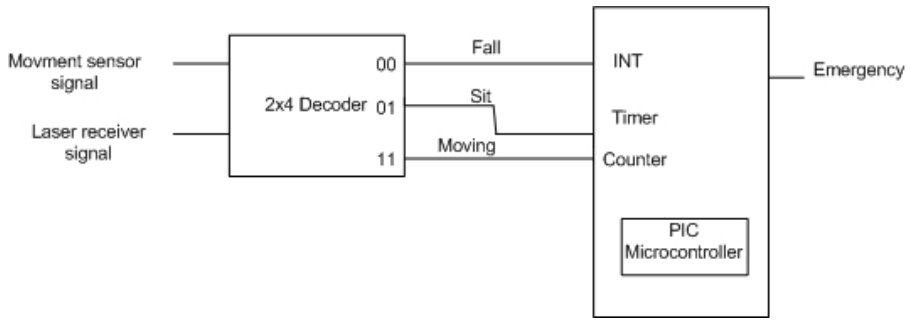


Figure 5.51

2- Actuators:

Actuators are the devices that give the smart home nice capabilities. A dispenser for taking the medicine was designed. The design description and implementation of the dispenser is discussed in this part.

1- Stepper Motor:

A stepper motor is a brushless, synchronous electric motor that can divide a full rotation into a large number of steps. It is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied

- Operation of the Stepper Motor:

Stepper motors operate differently from normal DC motors, which simply spin when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple "toothed" electromagnets arranged around a central metal gear. To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So, when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those slight rotations is called a "step." In that way, the motor can be turned a precise angle.

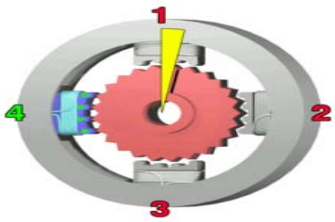


Figure 5.52

Figure1: the Teeth is attracted to the Proper Direction

There are two basic arrangements for the electromagnetic coils: bipolar and unipolar.

a- Unipolar motor:

In a unipolar stepper motor, there are four separate electromagnets. To turn the motor, first coil "1" is given current, then it's turned off and coil 2 is given current, then coil 3, then 4, and then 1 again in a repeating pattern. Current is only sent through the coils in one direction; thus the name unipolar.

A unipolar stepper motor will have 5 or 6 wires coming out of it. Four of those wires are each connected to one end of one coil. The extra wire (or 2) is called "common." To operate the motor, the "common" wire(s) is(are) connected to the supply voltage, and the other four wires are connected to ground through transistors, so the transistors control whether current flows or not. A microcontroller or stepper motor controller is used to activate the transistors in the right order. This ease of operation makes unipolar motors popular with hobbyists; they are probably the cheapest way to get precise angular movements.

b- Bipolar motor:

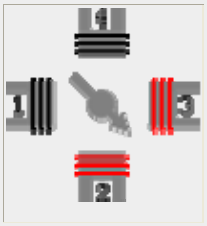
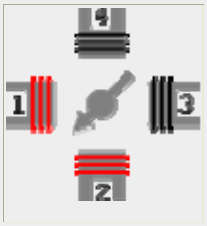
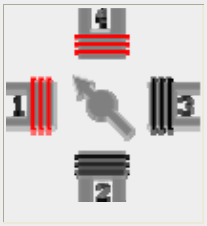
There are only two coils, and current must be sent through a coil first in one direction and then in the other direction; thus the name bipolar. Bipolar motors need more than 4 transistors to operate them, but they are also more powerful than a unipolar motor of the same weight. To be able to send current in both directions, H-bridge can be used to control each coil or a step motor driver chip.

A step motor can be viewed as a DC motor with the number of poles (on both rotor and stator) increased, taking care that they have no common denominator. Additionally, soft magnetic material with many teeth on the rotor and stator cheaply multiplies the number of poles (reluctance motor). Like an AC synchronous motor, it is ideally driven by sinusoidal current, allowing a stepless operation, but this puts some burden on the controller. When using an 8-bit digital controller, 256 microsteps per step are possible. As a digital-to-analog converter produces unwanted ohmic heat in the controller, pulse-width modulation is used instead to regulate the mean current. Simpler models switch voltage only for doing a step, thus needing an extra current limiter: for every step, they switch a single cable to the motor. Bipolar controllers can switch between supply voltage, ground, and unconnected. Unipolar controllers can only connect or disconnect a cable, because the voltage is already hard wired. Unipolar controllers need center-tapped windings.

It is possible to drive unipolar stepper motors with bipolar drivers. The idea is to connect the output pins of the driver to 4 transistors. The transistor must be grounded at the emitter and the driver pin must be connected to the base. Collector is connected to the coil wire of the motor.

Stepper motors are rated by the torque they produce. Synchronous electric motors using soft magnetic materials (having a core) have the ability to provide position holding torque (called *detent torque*, and sometimes included in the specifications) while not driven electrically. To achieve full rated torque, the coils in a stepper motor must reach their full rated current during each step. The voltage rating (if there is one) is almost meaningless.

The motors also suffer from EMF, which means that once the coil is turned off it starts to generate current because the motor is still rotating. There needs to be an explicit way to handle this extra current in a circuit otherwise it can cause damage and affect performance of the motor.

Step	Coil 4	Coil 3	Coil 2	Coil 1	Rotation
1	**ON**	**ON**	off	off	
2	Off	**ON**	**ON**	off	
3	Off	off	**ON**	**ON**	
4	**ON**	off	off	**ON**	

- Stepper Motor Modes Operations:
There are two modes the stepper motor can work with:

a- Full Step Mode:
The full step mode provides the maximum low speed torque because two windings are always energized. It is also provides the largest amount of rotation per step pulse. It will always be the noisiest acoustically, and has the highest mechanical torque ripple.

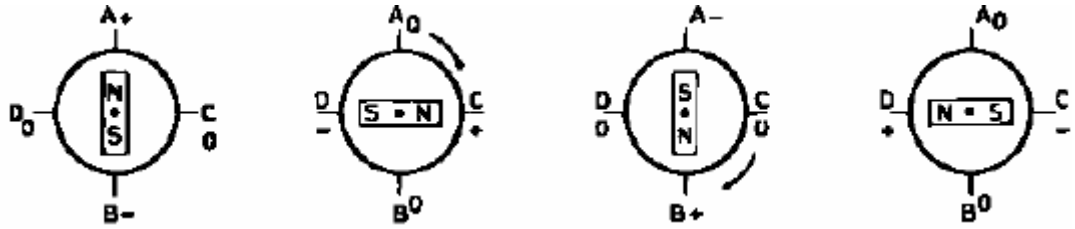


Figure 5.53 (Operation of theFull-Step)

b- Half Step Mode:

The half step mode normally provides the smoothest mode of operation. It is also provides the smallest amount of rotation per step pulse. Its principle advantage is a much higher resistance to mechanical motor and system resonance. Mostly for this reason, higher motor angular rotation speeds are usually possible with the half step mode.

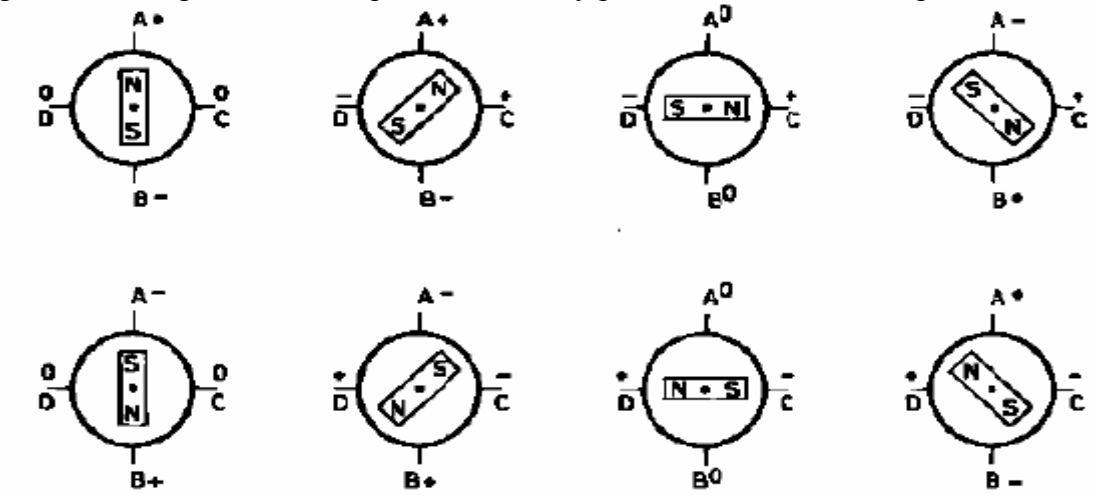


Figure 5.54 (Operation of the Half-Step)

The torque produced by a stepper motor depends on several factors.

1. The step rate
2. The drive current in the windings
3. The drive design or type.

c- H-Bridge (L298N):

The L298 Stepper Motor Controller is primarily intended for use with an L297N or L293E bridge driver in stepper motor driving applications. It receives control signals from the system's controller, usually a microcomputer chip, and provides all the necessary drive signals for the power stage. Additionally, it includes two PWM chopper circuits to regulate the current in the motor windings. With a suitable power actuator the L298 drives two phase bipolar permanent magnet motors, four phase unipolar permanent magnet motors and four phase variable reluctance motors. Moreover, it handles normal, wave drive and half step drive modes. The L298N contains two bridge driver stages, each controlled by two TTL-level logic inputs and a TTL-level enable input. In addition, the emitter connections of the lower transistors are brought out to external terminals to allow the connection of current sensing resistors. For the L298N STMicroelectronics' innovative ion-implanted high voltage/high current technology is used, allowing it to

handle effective powers up to 160W (46V supply, 2A per bridge). A separate 5V logic supply input is provided to reduce dissipation and to allow direct connection to the L297 or other control logic. The Two enable inputs are provided to enable or disable the device independently of the input signals.

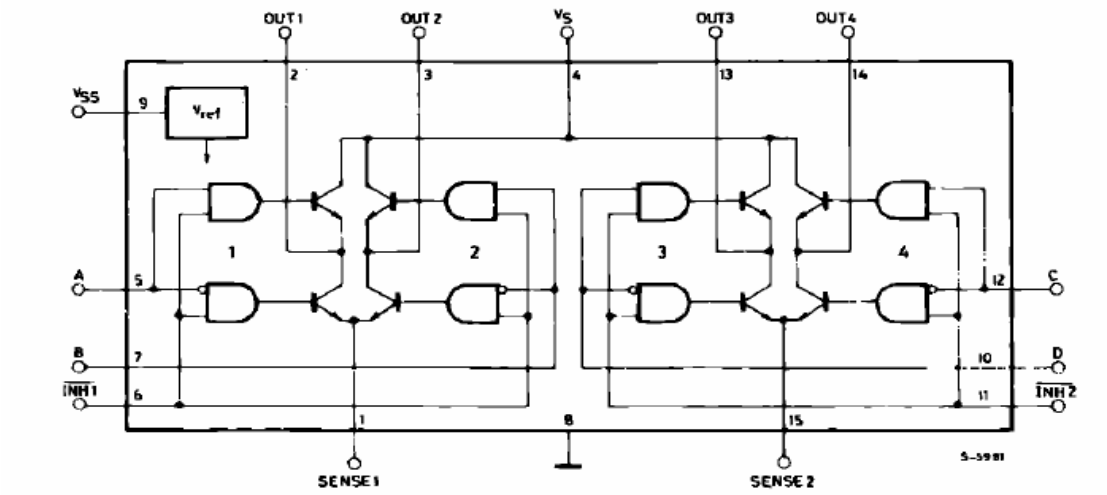


Figure 5.55 (The L298N Bridge)

2- The Dispenser Design:

a- The Controller Device:

PIC16F628 Microcontroller is used in the design as our device controller. The Microcontroller is responsible for controlling the rotation of the stepper motor through the H-bridge. Four bits of the PIC16F628 of PORTB is used for sending the signals that are controlling the stepper motor. These 4-bits are the input for the driver (H-Bridge) chip. The mode of the motor that is used is the half step. In order to understand the operation of the 4-bits with the stepper motor see the figure below:
The PIC microcontroller operates at 5V with 32.768 KHz as our oscillator. There is a speaker used in the design as an alarm for the medicine. It works when the time of the medicine comes.

b- The Driver Device:

The dual full bridge driver (L298N) is used in the design for deriving the motor. It locates in the middle between the microcontroller and the stepper motor. It has four inputs and four outputs. The four input pins are used to receive the signals from the microcontroller; however, the four outputs are connected to the four wires of the stepper motor. Each output should be connected to two diodes, as it shown in the logical design. So, the four outputs each with two diodes end up with 8-diodes. These eight diodes are used to avoid reversing any current to opposite direction for protecting the driver chip from damaged. There are two voltages connected to the L298 driver, one is 5V and the other is 12V. The 5V is used for the PIC microcontroller, but the 12V is used for the motor. The two ENABLE pins are connected to the 5V.

c- The Design Attributes:

- There are two power supplies used in the design.
  - 5V for the PIC microcontroller and the L298 driver
  - 12V for the stepper motor and the L298 driver as well.
- Four resistors with value of 470ohm are connected with the four inputs of the L298 driver.
- The two Current Sensing pins are connected to the ground with four resistors connected in parallel to distribute the current between them.
- The H-bridge chip is surrounded by a heater sink to drop out any extra heat accommodating on the chip.
- The mode that is used in the design is the half mode operation.

d- The Logical Design of the Dispenser:

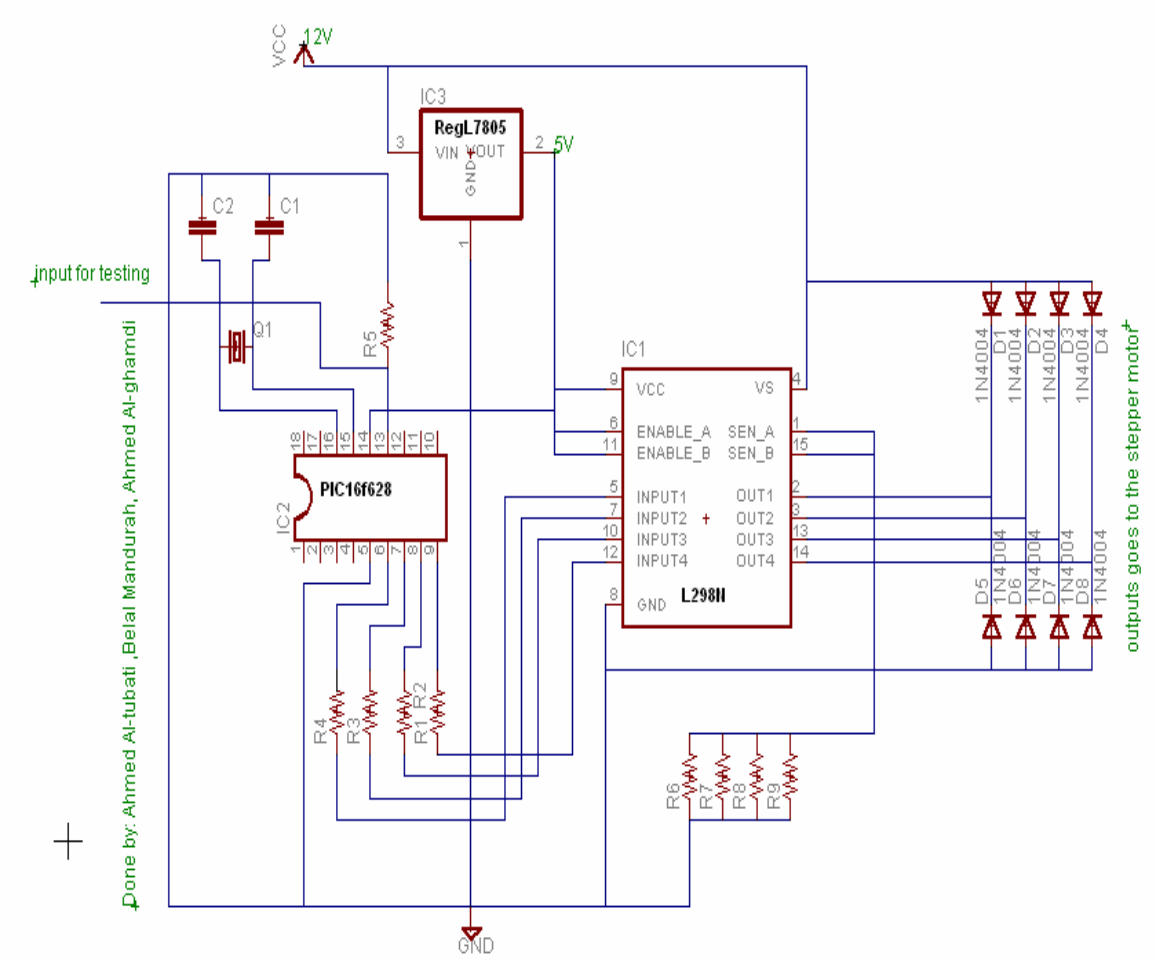


Figure 5.56 (The Logical Design of the Dispenser)

e- The Mechanical Design of the Dispenser:
By using the SolidWork Software, we implemented the mechanical design of the dispenser, including the dimension and the length of the dispenser.

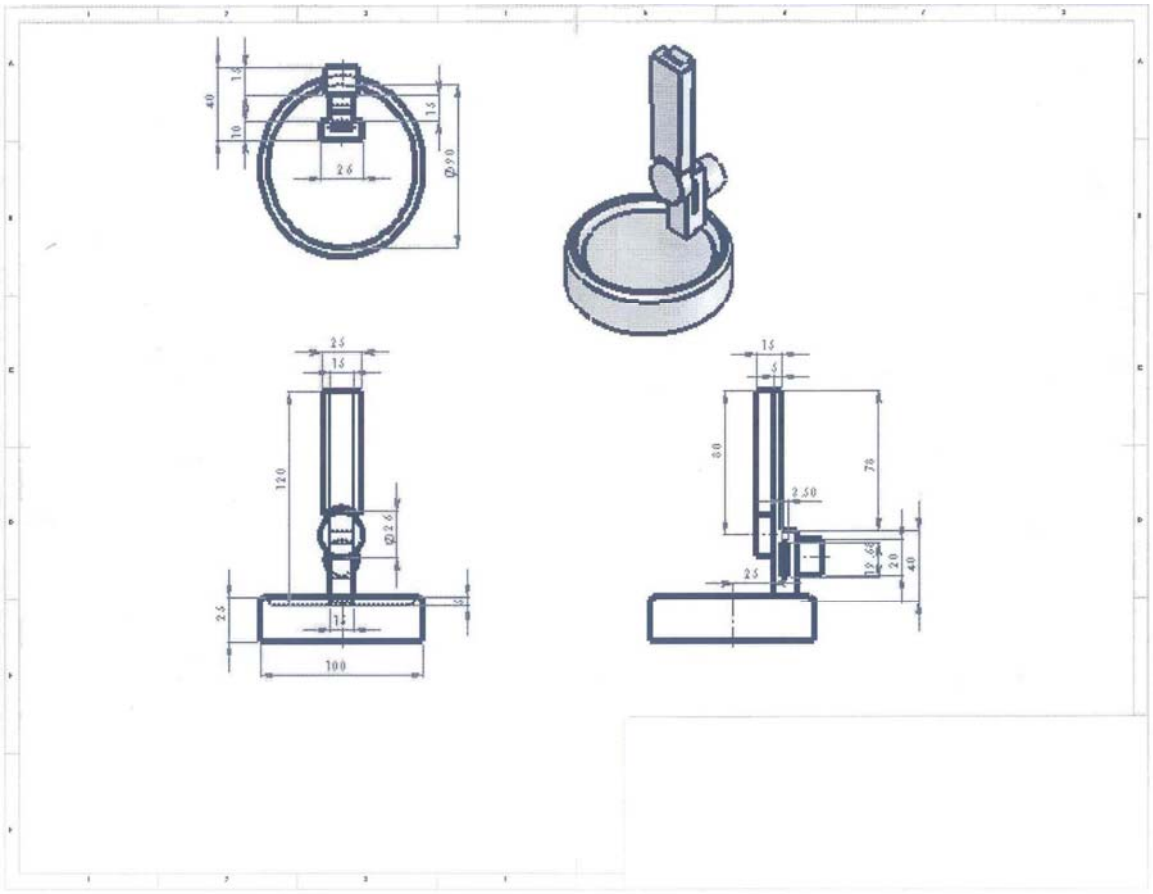


Figure 5.57 (The Mechanical Design of the Dispenser)

3- Multimedia:

Multimedia part of the hardware contains the LCD display and the camera. The design and implementation of these two devices is discussed in the following part.

a- LCD Display:
This project was divided into two parts. 1st part was to design the LCD display using Parallel port and Micro Controller, 2nd was to program the LCD and connect to main database in Elderly Home. It was required to design the circuit for the LCD and microcontroller to be compatible with parallel port and the alphabetical display and connect them together to receive the data from the PC. A program was written to control the parallel interface so it can be used in the integration part to give signals from the software to the hardware. The program was written using C language to show the time and the type of the medicine.

1- Parallel Port

The Parallel Port is the most commonly used port for interfacing homemade projects. This port will allow the input of up to 9 bits or the output of 12 bits at any one given time, thus requiring minimal external circuitry to implement many simpler tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. This will be a serial RS-232 port and thus, is a totally incompatible port.

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

The above table uses "n" in front of the signal name to denote that the signal is active low. e.g. n Error. If the printer has occurred an error then this line is low. This line normally is high, should the printer be functioning correctly. The "Hardware Inverted" means the signal is inverted by the Parallel card's hardware. Such an example is the busy line. If +5v (Logic 1) was applied to this pin and the status register read, it would return back a 0 in Bit 7 of the Status Register. The Parallel Port has three commonly used base addresses. These are listed in the next table below.

The 3BCh base address was originally introduced used for Parallel Ports on early Video Cards. This address then disappeared for a while, when Parallel Ports were later removed from Video Cards. They have now reappeared as an option for Parallel Ports integrated onto motherboards, upon which their configuration can be changed using BIOS.
LPT1 is normally assigned base address 378h, while LPT2 is assigned 278h. However this may not always be the case as explained later. 378h & 278h have always been

commonly used for Parallel Ports. The lower case h denotes that it is in hexadecimal. These addresses may change from machine to machine.

Address	Note
3BCh - 3BFh	Used for Parallel Ports which were incorporated in to Video Cards and now, commonly an option for Ports controlled by BIOS - Doesn't support ECP addresses.
378h - 37Fh	Usual Address For LPT 1
278h - 27Fh	Usual Address For LPT 2

- The printer port of a PC:
There are at least three versions of printer ports available now on PCs.
SPP (Standard Parallel Port), EPP (Enhanced Parallel Port) and ECP (Extended Capability Port)
The port can be controlled by using this program to send the bits

```
#include <stdio.h>
#include <dos.h>
void main(void)
{
unsigned int far *ptraddr; /* Pointer to location of Port Addresses */
unsigned int address; /* Address of Port */
int a;
ptraddr=(unsigned int far *)0x00000408;
for (a = 0; a < 3; a++)
{
address = *ptraddr;
if (address == 0)
printf("No port found for LPT%d \n",a+1);
else
printf("Address assigned to LPT%d is %Xh\n",a+1,address);
*ptraddr++;
}
}
```

Figure 5.58

2- Lcd Display:

Character (alphanumeric) type LCD modules are most commonly driven from an embedded microcontroller. It is sometimes desirable to be able to drive this type of module directly from a PC. This would allow the module to be tested or a demonstration or simulation to be set up quickly and with a minimum of engineering. This application note describes a method of driving an LCD character module from the printer port of a PC with minimal external hardware. To make this design as universal as possible it will be based on the least capable and oldest specification, SPP. To maintain compatibility with all versions of parallel ports now available on PCs only the output capabilities are used in this example. The LCD module provides a busy flag that can be read to control the timing of data and command writes. The timing of all data and command transfers is know and this is used to time the transfer of data and commands to the LCD in software. The parallel port has 12 buffered TTL output pins which are latched and can be written under program control using the processors out instruction. The port also has input pins but they are not used in this example. The suggested circuit is quite simple. The display, data and control lines are connected to the printer port lines. The read/write (RW) line is tied low to fix the display in the write mode. The Enable (E) and the Register Select (RS) lines are tied to two of the parallel port control lines. Most parallel ports have active or passive pull-ups on these lines, but not all. To be universal 10k pull-up resistors are shown on these two lines. Contrast control is provided by a 10k pot. DC power for the LCD and the back light, if installed, must be provided externally. LCDs require very little power to operate, typically less than 5mA. The outputs on the parallel port are able to source up to 10mA, so it is possible to power the LCD from one of the output lines. This doesn't apply to the back light which requires from 50mA to 300mA. One of the control lines can be used for this purpose as shown

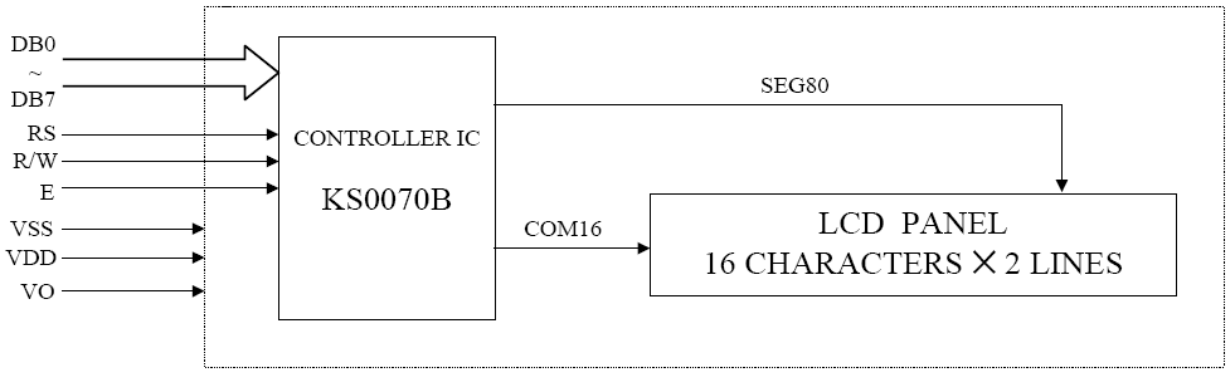


Figure 5.59

By using Control and Display Command and Standard Character Pattern, data can be sent easily (information is included in the appendix).

b- Camera:

This task is an implementation of an in-door camera the objectives are to monitor the elderly life without interference with her/his privacy.

- Design:
Camera implementations are a software and programming methods. The implementation is done using a Creative Webcam Vista Pro camera with a sensor enabled. It is used with its application and a design of a website.

- Description:
To monitor activities of the elder in the house, a camera is needed. To make this camera adapt with the smart environment, it has to be interfaced with the system with proper installation. It will keep track of events and will make it easier to monitor the elder if an emergency occurs. The main objective of the camera is to show the visitor who is standing outside the main door.

4- Microcontrollers:

Two types of microcontrollers are used in this project; Rabbit microcontroller and PIC microcontroller. The Rabbit is used to interface all the sensors that are distributed in the house and then sends the data received to the client/server application through wired LAN. The PIC microcontroller is used to interface the medical sensors and the falling sensor. The PIC uses MAX 232 to interface the sensors to the PC. Each of these microcontrollers and how they are interfaced with the sensors will be discussed.

1- Rabbit Microcontroller:
The rabbit Microcontroller is the mid point between the software (application and data base) and the hardware. The Rabbit function is to handle the signals coming from all of the sensors around the house and then send this information to the application using the Ethernet protocol to take the proper action.

To handle large number of sensors, parallel input / serial output shift register is used. The advantage of this technique is to reduce the number of pins used in the Rabbit to only 3 pins. One pin for the clock and one for the shift / load and the third for the input data.

- MM74HC165 Description:

This 8-bit serial shift register shifts data from QA to QH when clocked. Parallel inputs to each stage are enabled by a low level at the SHIFT/LOAD input. Also included is a gated CLOCK input and a complementary output from the eighth bit.. Parallel loading is inhibited as long as the SHIFT/LOAD input is high.

When taken low, data at the parallel inputs is loaded directly into the register independent of the state of the clock.

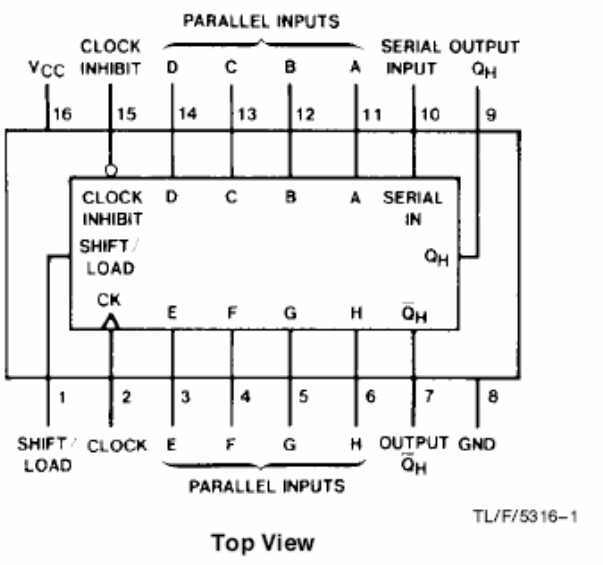


Figure 5.60

When the SHIFT/LOAD is LOW: the data will be loaded to the register.

When the SHIFT/LOAD is HIGH: no new data will be loaded to the register.

With every rising edge of the clock: the data stored in the register will be shifted from QA to QH. The connection of every sensor is shown below:

Water Sensor => H

Gas Sensor => G.

Light Sensor => F.

Pressure Sensor => E

Temperature Low Sensor => D

Temperature High Sensor => C

Door Sensor => B

Refrigerator's door Sensor=> A

SHIFT/LOAD is connected to the Pin A0 of the Rabbit.

Clock is connected to the pin A1 of the Rabbit.

Serial Output is connected to the pin B0 of the Rabbit.

The Pseudo code of the Rabbit is as follows:

Shift / load initially = 0

Force Shift / load = 1

Force the clock = 1

Read the coming data
If = 0 ➔ no event for this sensor
If = 1 ➔ an event is detected and the number of that sensor will be sent

Force clock = 0

Force clock = 1 // to read the next sensor

// same scenario is repeated with every interfaced sensor.

2- PIC Microcontroller:
PIC stands for Programmable Interface Controller or Peripheral interface controller. The PIC16F628 is used in the design. The PIC16F628 is from a family of low cost, high performance, CMOS and 8-bit microcontroller. It has features of power saving mode, watchdog timer with independent of oscillator for reliable operation and low voltage programming.

The features of the PIC family, especially PIC16F628, are integrated to reduce external components; so it will reduce system cost, enhancing system reliability and reducing power consumption. The sleep (power-down) mode offers power saving.

Memory is part of the microcontroller whose function is to store data. There are three types of memory provided by PIC16F628:

a- EEPROM is used for a periodic updating of data and the data are not lost when power is removed**.**

b- Regular RAM is used for temporary storage of data. But, data here is lost when power is removed.

c- Flash Memory is a program memory for storing a written program. The program memory is made in Flash technology which gives the advantage to be programmed and cleared more than once.

In order to program the PIC a compiler is needed, programmer and a programming circuit. The IC PROG compiler is used. The MPLAB is used to convert assembly codes to hexadecimal to download it to the chip. The chip of the microcontroller contain essential elements such as CPU, memory unit, Buses, Input/Output ports, timer unit, Analog to Digital converter and watchdog.

- MAX232:

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The MAX232 chip is responsible for serial connection.

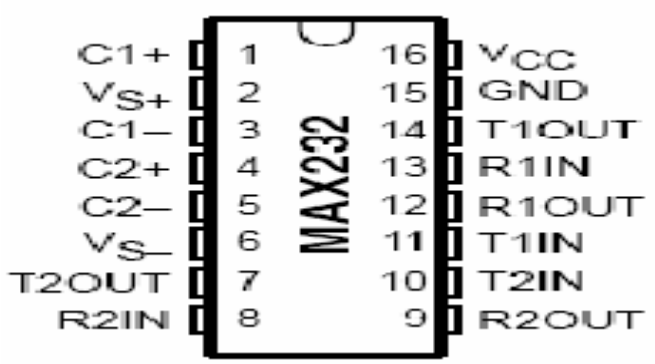


Figure 5.61 (AX232 chip)

- **Function Tables of MAX232**

For the Driver

INPUT Tin	OUTPUT Tout
Low	High
High	Low

For the Receiver

INPUT Tin	OUTPUT Tout
Low	High
High	Low

- Medical Sensors Interface with PIC Microcontroller:

The medical sensors are interfaced with PIC microcontroller (PIC16F628A) to update the database with new events. Once the event is occurred, the PIC microcontroller sends a signal via serial port (RS232) to the PC (Personal Computer). The PIC is responsible only for the sensors that are integrated with wheel chair (Medical Sensors).

a- The Design:

The PIC microcontroller is designed to have 14 sensors connected with. Since there are 2 Ports in the PIC microcontroller (PORTA & PORTB) each with 8-bits, all the 8-bits of port A are being used to hold sensors plus 6-bits of port B. So, the total number of sensors that may be able to be connected to the PIC is 14 sensors. The 8-bits of port A are ST (Schmitt Trigger). But, the port B is either ST or TTL (Transistor-Transistor Level). The 2-bits of port B that are not used for sensors, they have been used for the serial port with MAX232. One of them is used for the transmission mode and the other for the receiving mode.

The MAX232 is used to connect the PIC microcontroller with serial port (RS232). Since we need only to transmit the signals to the database, we used only the transmission pin in MAX232. No need for the receiving pin to be connected.

b- How the design works:

The PIC microcontroller receives the signals from the sensors through either port A or port B. After the signals are received, The PIC microcontroller sends them to the serial port via MAX232 chip then to the software that is running on the PC. The software will handle the signals and process them. Three registers of The PIC microcontroller are being used to specify each sensor. The PIC microcontroller sends three bytes each time the event is occurred. The first byte is 00 always to separate between events. The 00 byte is sends at the first of each event. Then the other 2 bytes is sent. Each bit in the last 2 bytes is assigned to a particular sensor. For example see the following table

Byte	Bits	Event
80	1000 0000	Sensor A when 7^{th} bit is occurred
40	0100 0000	Sensor B when 6^{th} bit is occurred
02	0000 0010	Sensor C when 2^{nd} bit is occurred

There are three modes should be configured between the PIC microcontroller and the software program:

1. Configuration mode.
2. Receiving mode
3. Transmission mode.

The PIC microcontroller needs only the configuration mode and the transmission mode. But the software needs the configuration mode and the receiving mode.

The asynchronous mode used in the design. The data rate is 2.4Kbps with no parity bit.

c- The Logical Design:

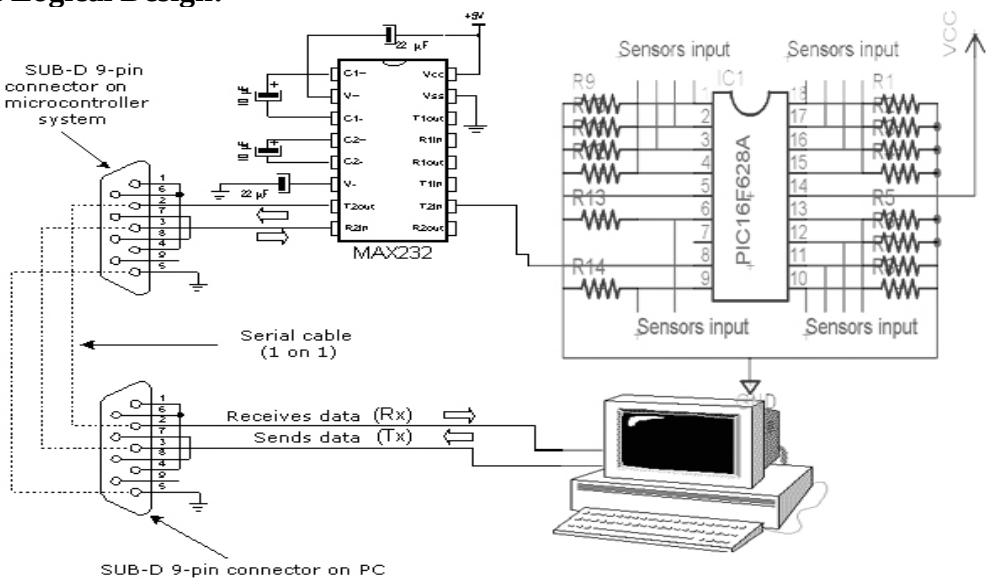


Figure 5.62 (Logical Design)

d- Fall Sensor Interfacing:

The PIC microcontroller will dealing with these signals and it will send these signals to PC as follows:

- Three registers are used to distinguish between sensors. (Each bit inside the registers will be points to particular sensor).

- Three modes should be enabled to communicate with RS232 and they are:
  i. Configuration mode.
  ii. Receive mode.
  iii. Transmit mode.

- The PIC microcontroller sends 2 bytes to the PC. The first byte is 00 and the second byte is divided in to 2 parts the most significant bits and the least significant bits. The most significant bits will be 00 and the least significant bits will be showed in the following table:

Input	Logic	Output
Move' & Laser'	00	Fall
Move' & Laser	01	Sit/Still
X "Don't care"	10	X
Move & Laser	11	Move

Because only the falling sensor is used, 2 bits are used and can increase the number of sensors that the PIC can deal with, by modifying the program. It can deal with up to 16 sensors simultaneously.

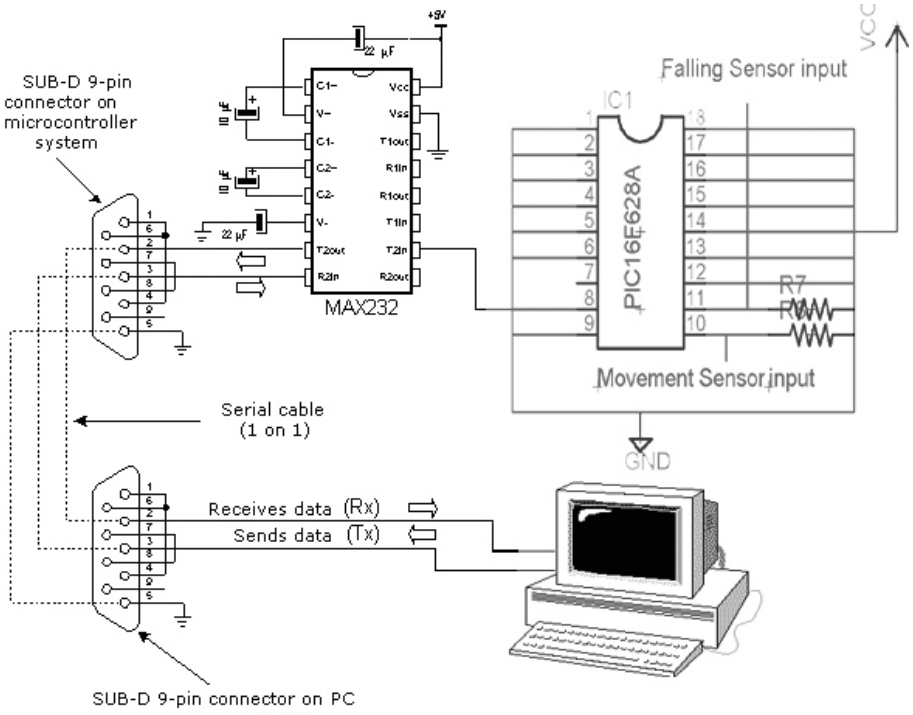


Figure 5.63

To communicate with the PIC Microcontroller through the serial port, this application was developed. The application reads the massage sent by the microcontroller to the PC and then interfaces it with the client/server application.

- Description:
Using the VB.NET Programming Language, because it is more portable, flexible, and secure, the following code was written to read the massage sent by the microcontroller to the PC. Then, the code interprets the massage which is in hex format to a binary format that can be then interpreted as signals to the flagged interrupts by the sensors. After that, the name of the sensor is written to a text file to use it by the communication team and the database team. The program interface is showing the interrupted sensors for further assistance. The code is tested and it was free of errors. This task took two weeks.

The following figure shows the main interface of the program:

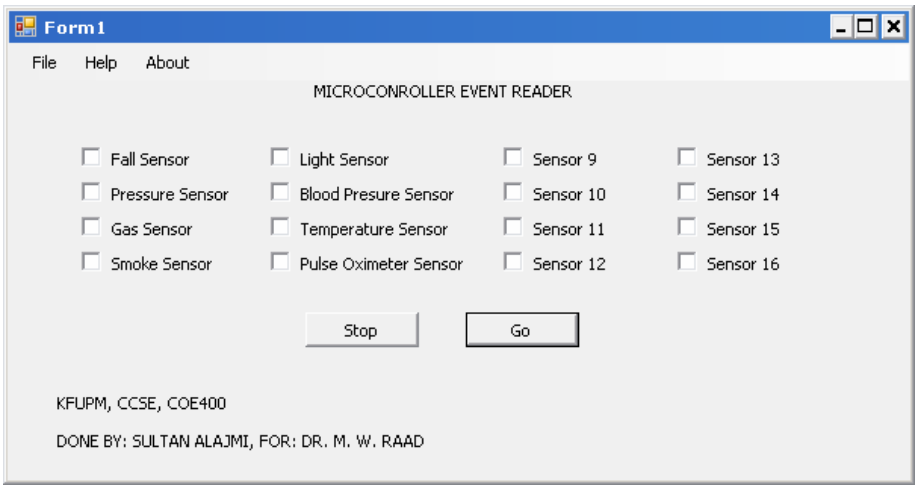


Figure 5.64

The program can show until 16 sensors at a time.

The following figure shows the text file if the fall sensor is triggered:

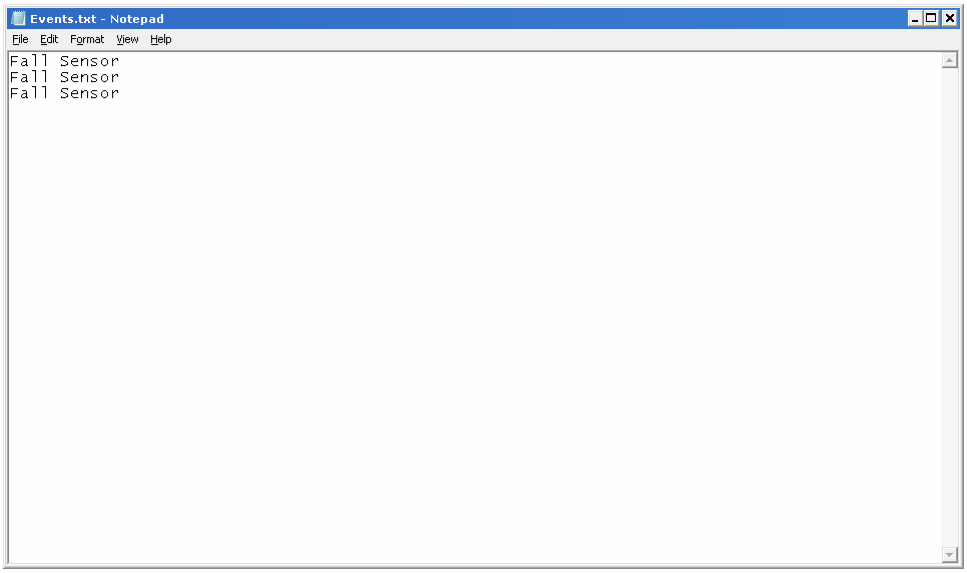


Figure 5.65

VI- ARCHITECTURAL DESIGN

In order to do a layout design of the house that is going to be used for the elder person in the project. Regular meetings with friends majoring in Architecture Engineering in University of Bahrain were made. The project goals were presented and gave them an overview in order for them to grasp what is specifically needed. The house was designed with specific restrains in mind like putting the toilet near the bedroom and like making the living room open on the kitchen for ease of getting food. All of that on a limited area which is just perfect according to the life of the elderly and even cost wise.

Figure 6.1

Building the Model:

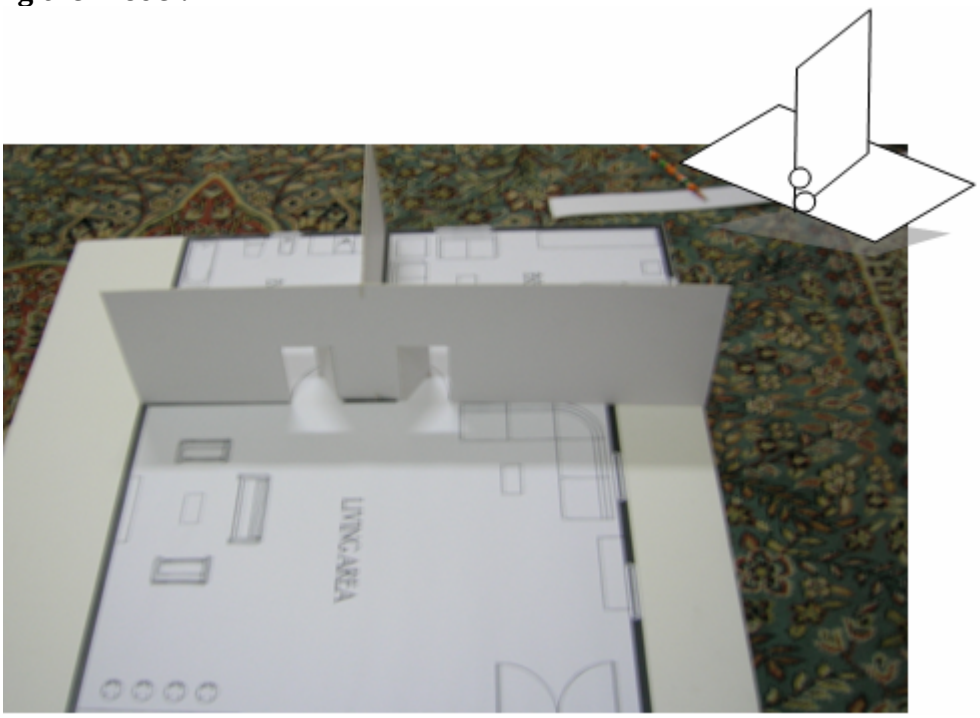


Figure 6.2

As a company, the need to market our product is of extreme importance. A touch of creativity is added. A fully portable model than needs around one minute to set it up and by the end of the presentation of a probable invest is made, the whole model folds in and transforms into an elegant briefcase shape. It solved the problems of space of storage and portability for presentations.

VII- FEASABILITY OF IMPLENTATION

The smart home was designed so that it can satisfy the requirements of the design as much as possible. Security was implemented in all of the areas of the home so that it can ensure the safety of the elder anywhere and anytime. Alerting the doctor was also an important issue of the design. Using the web application, it will make it easier to follow up the actions of the elder all the time.

CONCLUSION

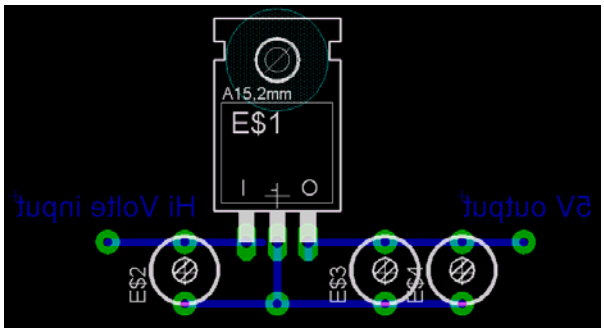
Designing a complete system is a huge part of computer engineering. It made us taste the flavor of designing and implementing a system that works together. It also gave us inspiration in coming up with creative ideas and implement these ideas in a real life application. This project gave us experience in how to work in a team with one goal and one accomplishment.

APPENDIX

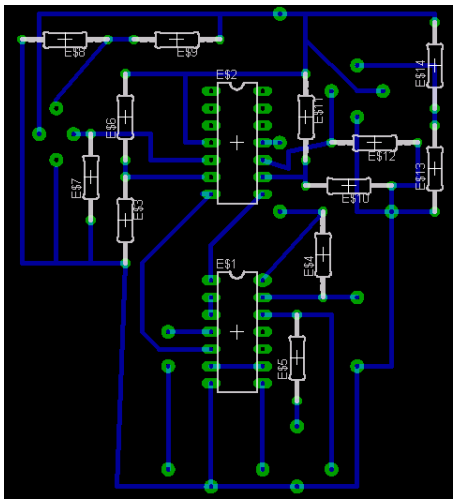
1- PCB Fabrication:

It was not done because of the person who is responsible to print it is busy. Design was made to make it as small as possible to save space on the board to reduce the cost. This was done by belal mandourah.

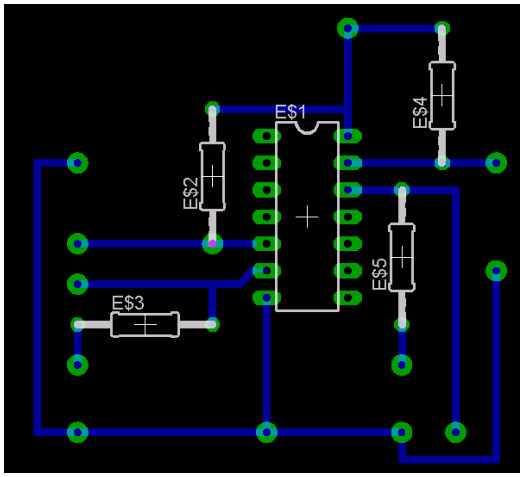
* 5V Regulator:



* Emergency Button and Gas Sensor circuit provided by Hessian Naji and Ahmed Aseri:

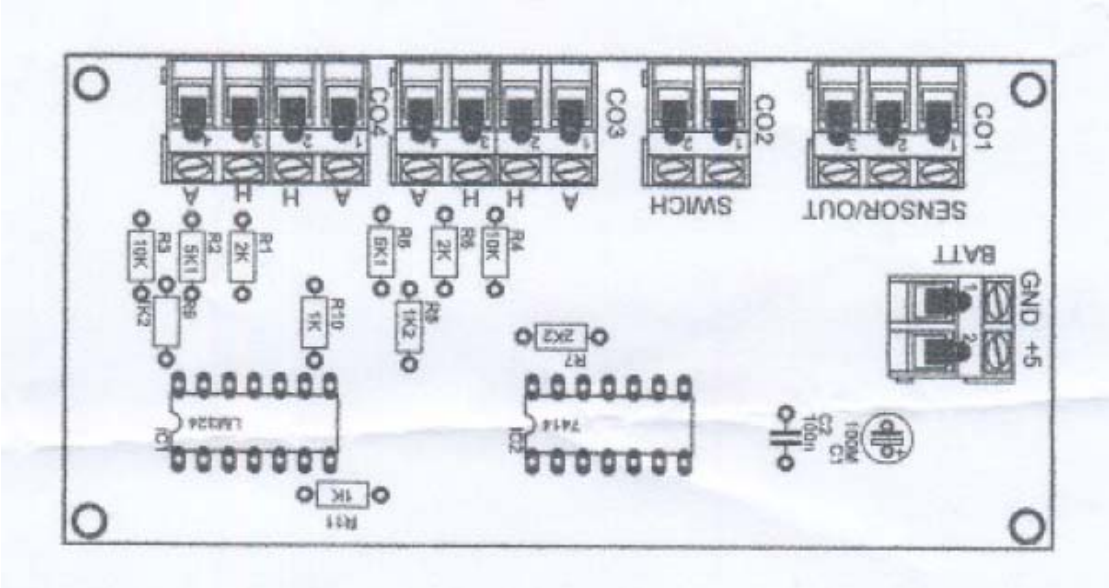


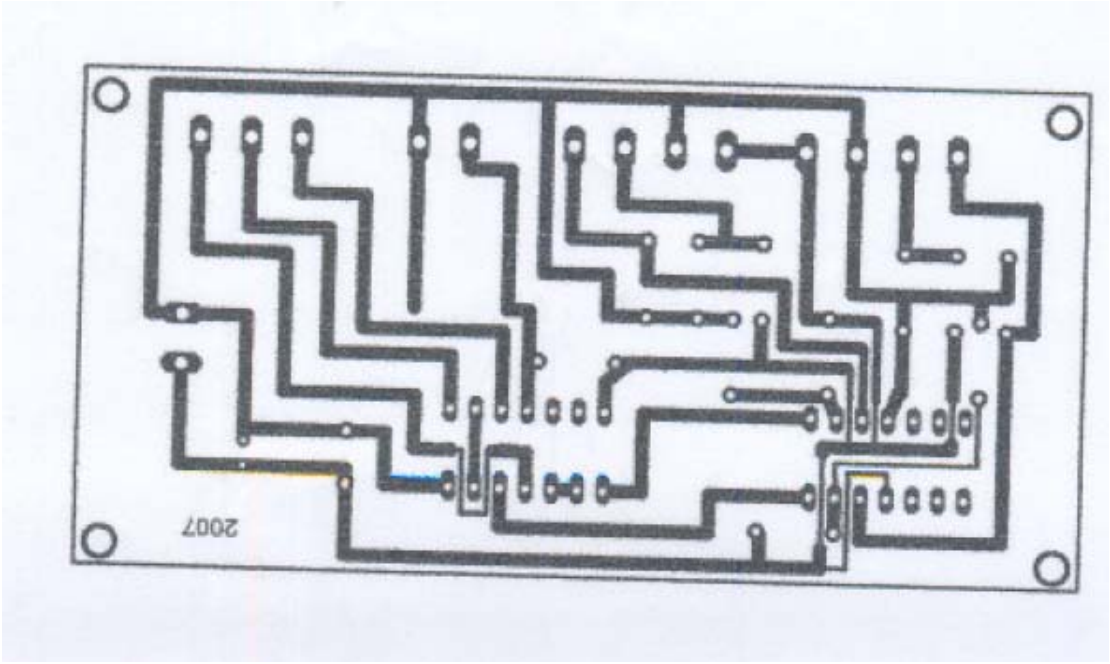
* Reading Switch and Door Circuit provided also by Husain and Ahmed



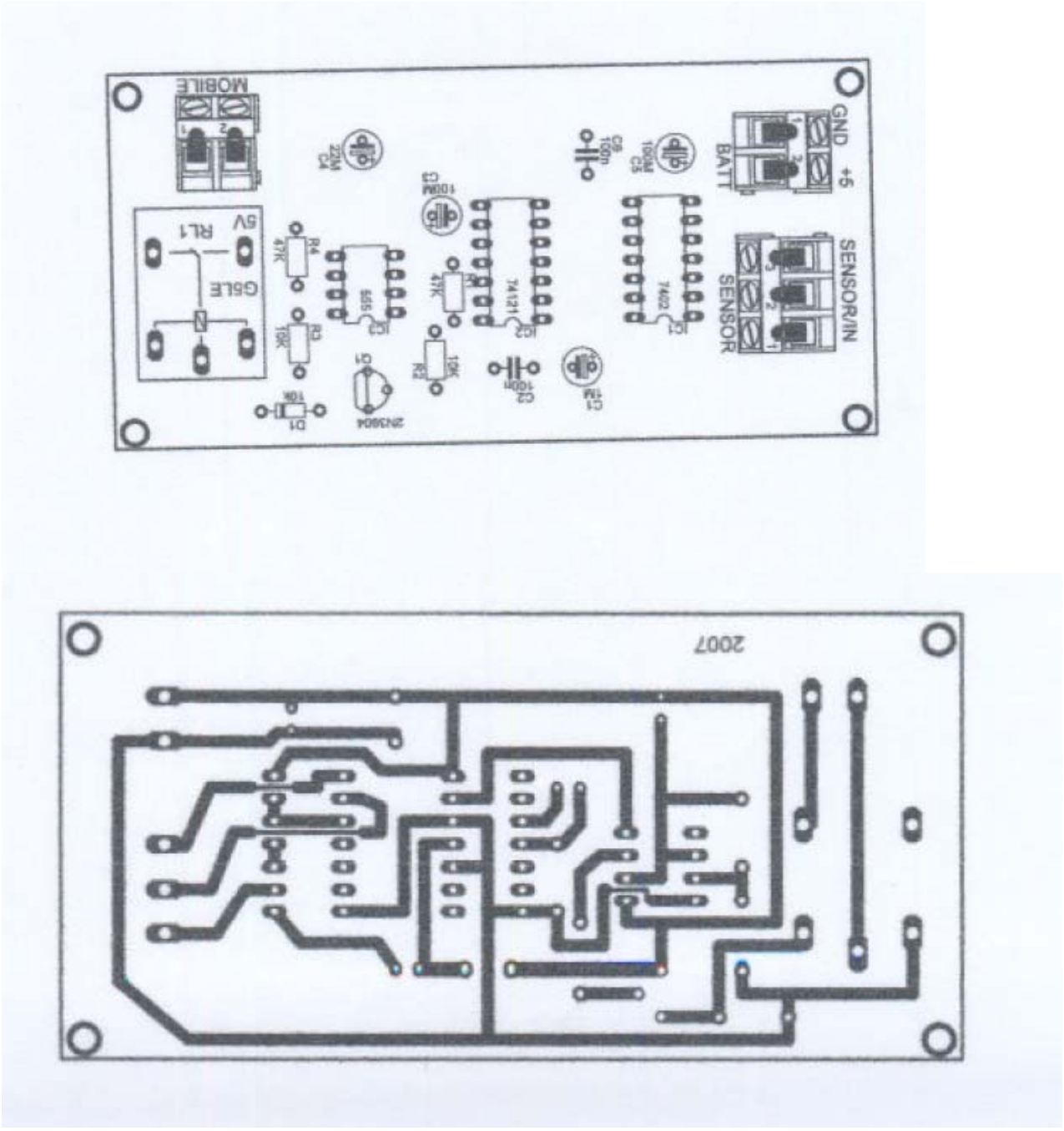
Medical Sensor PCB:

-Emergency button, co and h sensors

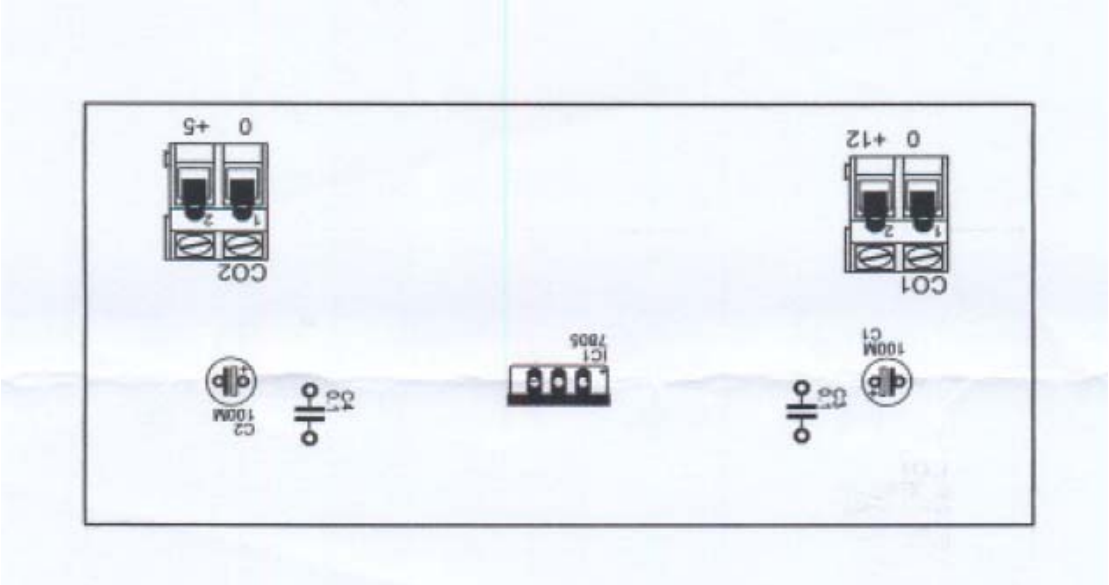




-Nor gate, Monostable_7412, Astable_555 and Relay



-12 V to 5V Power Supply



2- Smart Card Codes:

Code for the smart card application:

- Part of the Methods to get the Name and ID

```
//*****************************<Get the Name>************************
                smartCardObject.setcommand(commandGetName);
                smartCardObject.exchange(); // communicate
                test.Text = "3";
                Byte[] bName = new Byte[20];
                for (short j = 0; j < 20; j++)
                    bName[j] = smartCardObject.getresponsebyte(j);
                test.Text = "4";

                label5.Text =
System.Text.ASCIIEncoding.UTF8.GetString(bName);
                test.Text = "5";
  //***************************<Get ID>*********************
                //card.smartCardObject.connectpcsc(contactLess);
                //card.SelectApplication(applicationName);
                //*********
                smartCardObject.setcommand(commandGetID);
                test.Text = "6";
                smartCardObject.exchange(); // communicate
                test.Text = "7";
                Byte[] bID = new Byte[20];
                for (short j = 0; j < 20; j++)
                    bID[j] = smartCardObject.getresponsebyte(j);
                test.Text = "8";

           label6.Text = System.Text.ASCIIEncoding.UTF8.GetString(bID);

-----------------------------------------------------------------------
```

- Login Page (for smart card application)

```
namespace SmartCardClientApplication
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            error.Text = "";
            TextReader usps = new StreamReader("UserPass.txt"); // read
the file from text and then compare it
            string user = textBox1.Text;
            string pass = textBox2.Text;
            if (user == usps.ReadLine().ToString() && pass ==
usps.ReadLine().ToString())
            {
```

```
                usps.Close();
                textBox1.Clear();
                textBox2.Clear();
                Form2 frm = new Form2();
                frm.ShowDialog();
            }
            else
            {
                textBox1.Clear();
                textBox2.Clear();
                error.Text = "The password or the user name is not
correct, please try again";
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            error.Text = "";
        }
    }
}
```

- Database connectivity:

```
//----------------------Connection to database--------------------
        string connectionString = @"Initial Catalog=Warehouse;" +
        @"Data Source=alderydb\wdbserver1;" +
        @"User ID=sa;" +
        @"Password=coe400";

        dbConnection = new SqlConnection(connectionString);

            try
            {
                dbConnection.Open();
                label18.Text = "DB OK";
            }
            catch (Exception)
            {
                err.Text = "Error connecting to the database!";
                return;
            }

            SqlCommand myCommand = new SqlCommand("SELECT * FROM
EMPLOYEE WHERE name=\'EMPLOYEE\' AND emp_id=\'" /*+ empID + "\'"*/,
dbConnection);

            SqlDataReader reader = myCommand.ExecuteReader();
            Boolean result = reader.Read();
            reader.Close();
            if (result)
            {
                label10.Text = "Approved-Door Open";
            }
            else
                label10.Text = "Unauthorized person";
```

- **Log file (for the Wheelchair):**

```
public void ReceiveMethod()
        {
            while (true)
            {
                string strLine;

                try
                {
                    FileStream aFile = new FileStream("data.txt",
FileMode.Open);
                    StreamReader sr = new StreamReader(aFile);
                    strLine = sr.ReadLine();

                    while (strLine != null)
                    {
                        //strLine = sr.ReadLine();
                        listBox1.Items.Add("Data Sent:" + strLine);
                        writer.WriteLine (Int32.Parse(strLine));
                        writer.Flush();
                        strLine = sr.ReadLine();
                    }
                    aFile.Close();
                    sr.Close();
                }
                catch (IOException e)
                {
                    Console.WriteLine("An IO exception has been
thrown!");
                    Console.WriteLine(e.ToString());
                    return;
                }
                Thread.Sleep(5000);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            ReceiveMethod();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

    }
}
```

Java Code

```
// Smart Card Application (Husain Al-Subaii ID#918572)
// COE 400 (062)
import com.multos.ifd.*;
import com.multos.framework.*;
public final class SMART1
{
 static final int ERR_UNDERFLOW = 0x6906;

 // Commands understood by this application.
 static final int CMD_SETPID = 0x20;
 static final int CMD_GETPID = 0x30;
 static final int CMD_GETRNAME = 0x44;
 static final int CMD_SETPNAME = 0x42;
 static final int CMD_SETREL = 0x43;
 static final int CMD_GETREL = 0x45;

 // APDU class this application responds to.
 static final int MYAPP_CLA = 0x70;

 // Personale data: pname.
 static byte rname[] =
{(byte)'H',(byte)'u',(byte)'s',(byte)'a',(byte)'i',(byte)'n',0,0,0,0,0,0,0,0,0,0,0,0,0,0};

 // Personale data: lastnames.
 static byte relation[] =
{(byte)'P',(byte)'h',(byte)'y',(byte)'s',(byte)'i',(byte)'c',(byte)'i',(byte)'a',(byte)'n',0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

 // ID
 static byte id[] =
{(byte)'0',(byte)'0',(byte)'9',(byte)'1',(byte)'8',(byte)'5',(byte)'7',(byte)'2'};

 public static void main()
 {
   // Check CLA.
  if (APDU.getCLA() != MYAPP_CLA)
    MULTOS.exitSW(ISO7816.ERR_CLASS_NOT_SUPPORTED);

  // Get and decode INS.
  int BOB = APDU.getINS();
```

```
      switch(BOB)
      {

   case CMD_SETPID:
  // Ensure ISO case 3 command
  if (!APDU.checkCase(3))
    MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

  // Take new ID
    Util.arrayCopy(APDU.data, 0, id, 0, id.length);
   break;

case CMD_GETPID:
  // Ensure ISO case 2 command.
  if (!APDU.checkCase(2))
    MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

  // Report points in R-APDU.
   Util.arrayCopy(id, 0, APDU.data, 0, id.length);

  // Exit setting La.
  APDU.setLa(id.length);
  break;

 case CMD_GETRNAME:
  // Ensure ISO case 2 command
  if (!APDU.checkCase(2))
  MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

  // Copy from local surname and forename arrays to R-APDU data.
  Util.arrayCopy(rname, 0, APDU.data, 0, rname.length);

  // Set La to reflect data length.
  APDU.setLa(rname.length );
  break;

case CMD_GETRNAME:
  // Ensure ISO case 3 command
  if (!APDU.checkCase(3))
   MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

  // Copy surname and forename fields into permanent storage
  Util.arrayCopy(APDU.data, 0, rname, 0, rname.length);
   break;
```

```
        case CMD_SETREL:
        // Ensure ISO case 3 command
         if (!APDU.checkCase(3))
        MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

        Util.arrayCopy(APDU.data, 0, relation);
         break;

         case CMD_GETREL:
         // Ensure ISO case 2 command
        if (!APDU.checkCase(2))
        MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

       Util.arrayCopy(relation, 0, APDU.data, 0, relation.length);
       APDU.setLa(relation.length);

            break;


    }
  }


import com.multos.framework.*;
import com.multos.ifd.*;


public final class SMART
{
        static final int ERR_UNDERFLOW = 0x6906;

        // Commands understood by this application.
        static final int CMD_GETPID = 0x31;
        static final int CMD_GETRNAME = 0x32;
        static final int CMD_GETREL = 0x33;


        // APDU class this application responds to.
        static final int MYAPP_CLA = 0x70;

        // Personale data: pname.
        static byte rname[] = { (byte)'H', (byte)'a', (byte)'i', (byte)'t', (byte)'h', (byte)'a',
(byte)'m',(byte)'_', (byte)'S', (byte)'h', (byte)'u', (byte)'k', (byte)'r', (byte)'i', 0, 0, 0, 0, 0, 0
};
```

```
        static byte relation[] = { (byte)'D', (byte)'o', (byte)'c', (byte)'t', (byte)'o', (byte)'r', 0,
0, 0, 0, 0, 0, 0, 0, 0,0,0,0,0 };

        // ID
        static byte id[] = { (byte)'2', (byte)'3', (byte)'6', (byte)'1', (byte)'7',
(byte)'5',0,0,0,0,0 };


        public static void main()
        {

                // Check CLA.
                if (APDU.getCLA() != MYAPP_CLA)
                        MULTOS.exitSW(ISO7816.ERR_CLASS_NOT_SUPPORTED);

                //----------------------------------------

                // Get and decode INS.
                int ins = APDU.getINS();
                switch (ins)
                {



                        case CMD_GETPID:
                                // Ensure ISO case 2 command.
                                if (!APDU.checkCase(2))

        MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

                                // Report points in R-APDU.
                                Util.arrayCopy(id, 0, APDU.data, 0, id.length);

                                // Exit setting La.
                                APDU.setLa(id.length);
                                break;

                        //--------------------------------------------------

                        case CMD_GETRNAME:
                                // Ensure ISO case 2 command
                                if (!APDU.checkCase(2))

        MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);
```

```
                        // Copy from local surname and forename arrays to R-
APDU data.
                        Util.arrayCopy(rname, 0, APDU.data, 0, rname.length);
                        //Util.arrayCopy(lastnames, 0, APDU.data, pname.length,
lastnames.length);

                        // Set La to reflect data length.
                        APDU.setLa(rname.length);
                        break;

                    case CMD_GETREL:

                        if (!APDU.checkCase(2))

        MULTOS.exitSW(ISO7816.ERR_WRONG_LENGTH);

                        Util.arrayCopy(relation, 0, APDU.data, 0, relation.length);

                        APDU.setLa(relation.length);
                        break;


                }
        }
}
```

3- Rabbit Code:

```
#class auto
#define TCPCONFIG 1
#use "dcrtcp.lib"
#define CLOCK 0
#define SHIFT 2
#define INPUT 2
#memmap xmem
#define ADDRESS "10.80.16.66"    // destination address
#define PORT "9090"        // destination port number

main(){

   int gas;
   int light;
  int pressure;
   int templ;
  int temph;
  int water;
   int door;
   int ref;
   int motion;
   int goff;
   int loff;
   int poff;
   int thoff;
   int tloff;
   int woff;
   int moff;
   int droff;
   int reoff;
          int found;
          int count;
   int gp;                    // to store the signal of the pin that is detecated for gas
   int lp;      // to store the signal of the pin that is detecated for light
   int pp;     // to store the signal of the pin that is detecated for pressure
   int thp;     // to store the signal of the pin that is detecated for temp
   int tlp;
   int wp;
   int mp;
   int drp;
   int rep;

          int d1;
          int d2;
```

```
        int d3;
  int d4;
  longword host,seq;
  char  buffer[100];
  static tcp_Socket socket;
    // numbering the sensors
  gas = 1;
  light = 2;
        pressure = 3;
  templ = 51;
        temph= 5;
  motion = 6;
  door = 10;
  ref = 20;

   // Inabling Port A as an output
 WrPortI(SPCR,&SPCRShadow, 0x084);
 BitWrPortI(PBDDR,&PBDDRShadow, 0, INPUT );
 BitWrPortI(PADR, &PADRShadow, 0, SHIFT);

 BitWrPortI(PADR, &PADRShadow, 0, CLOCK);

    sock_init();
                while (ifpending(IF_DEFAULT) == IF_COMING_UP)
                {
                        tcp_tick(NULL);
                }

        if (!(host = resolve(ADDRESS))) {
                        puts("Could not resolve host");
        exit( 3 );
        }


while(1)
{

        BitWrPortI(PADR, &PADRShadow, 1, SHIFT);

                // Gas Sensor
   //*************************************************************
   BitWrPortI(PADR, &PADRShadow, 1, CLOCK); // force clock to One
                gp = BitRdPortI(PBDR ,  INPUT );      // read the pin status

                BitWrPortI(PADR, &PADRShadow, 0, CLOCK);
```

```
  if(gp == 0)
{
  goff = 0;

}
            if(gp == 1 && goff == 0)       // Test whether Gas pin is High or not
{
    tcp_open(&socket,0 , host, 9090 , NULL  );   // open the Socket
            // Wait for the interface to come up
d3=0;
while (d3==0)
            {
    d3 = sock_established( &socket );
                    tcp_tick(NULL);
    }

            sprintf(buffer , "%d", gas);

    sock_write(&socket, buffer,1);

      sock_write(&socket,buffer,1);

            sock_write(&socket,buffer,1);

            sock_write(&socket,buffer,1);

 sock_close(&socket );

    goff = 1;     // to make sure the prgram will not send the
}           // number again while it is still high

//*************************************************************
    // Light Sensor
 //*************************************************************
BitWrPortI(PADR, &PADRShadow, 1, CLOCK);
    for(d2 = 0 ; d2< 1000 ; d2++)
            {
            }

            lp = BitRdPortI(PBDR ,  INPUT );

            BitWrPortI(PADR, &PADRShadow, 0, CLOCK);

  if(lp == 0)
{
  loff = 0;
```

```
  }
if(lp == 1 && loff == 0)
  {
      tcp_open(&socket,0 , host, 9090 , NULL  );
              // Wait for the interface to come up
      d3=0;
  while (d3==0)
              {
      d3 = sock_established( &socket );
                      tcp_tick(NULL);
      }

              sprintf(buffer , "%d", light);

      sock_write(&socket, buffer,1);

        sock_write(&socket,buffer,1);

              sock_write(&socket,buffer,1);

              sock_write(&socket,buffer,1);

   sock_close(&socket );

      loff = 1;
  }


  //*************************************************************
    // Pressure Sensor
  //*************************************************************
BitWrPortI(PADR, &PADRShadow, 1, CLOCK);
                      pp = BitRdPortI(PBDR ,  INPUT );

              BitWrPortI(PADR, &PADRShadow, 0, CLOCK);
              for(d2 = 0 ; d2< 1000 ; d2++)
              {
              }
   if(pp == 0)
   {
     poff = 0;

   }
   if(pp == 1 && poff == 0)
   {
```

```
        tcp_open(&socket,0 , host, 9090 , NULL  );
                // Wait for the interface to come up
   d3=0;
  while (d3==0)
                {
        d3 = sock_established( &socket );
                        tcp_tick(NULL);
        }

                sprintf(buffer , "%d", pressure);

        sock_write(&socket, buffer,1);

          sock_write(&socket,buffer,1);

                sock_write(&socket,buffer,1);

                sock_write(&socket,buffer,1);

    sock_close(&socket );

      poff = 1;
   }


   //*************************************************************
           // Temperature low Sensor
 //*************************************************************
 BitWrPortI(PADR, &PADRShadow, 1, CLOCK);
                tlp = BitRdPortI(PBDR ,  INPUT );

                BitWrPortI(PADR, &PADRShadow, 0, CLOCK);
                for(d2 = 0 ; d2< 1000 ; d2++)
                {
                }
    if(tlp == 0)
    {
      tloff = 0;
    }
         if(tlp == 1 && tloff == 0)
    {
        tcp_open(&socket,0 , host, 9090 , NULL  );
                // Wait for the interface to come up
    d3=0;
   while (d3==0)
                {
```

```
    d3 = sock_established( &socket );
                    tcp_tick(NULL);
    }

            sprintf(buffer , "%02d", templ);

    sock_write(&socket, buffer,2);

      sock_write(&socket,buffer,2);

            sock_write(&socket,buffer,2);

            sock_write(&socket,buffer,2);

 sock_close(&socket );

  tloff = 1;
}

//*************************************************************
      // Temperature high Sensor
 //*************************************************************
    BitWrPortI(PADR, &PADRShadow, 1, CLOCK);
            thp = BitRdPortI(PBDR ,  INPUT );

            BitWrPortI(PADR, &PADRShadow, 0, CLOCK);
            for(d2 = 0 ; d2< 1000 ; d2++)
            {
            }
  if(thp == 0)
{
  thoff = 0;

}
if(thp == 1 && thoff == 0)
{
    tcp_open(&socket,0 , host, 9090 , NULL  );
            // Wait for the interface to come up
 d3=0;
while (d3==0)
            {
    d3 = sock_established( &socket );
                    tcp_tick(NULL);
    }

            sprintf(buffer , "%d", temph);
```

```
        sock_write(&socket, buffer,1);

          sock_write(&socket,buffer,1);

                  sock_write(&socket,buffer,1);

                  sock_write(&socket,buffer,1);

   sock_close(&socket );

      thoff = 1;
  }



    //*************************************************************
                  //  Door Sensor
    //*************************************************************
BitWrPortI(PADR, &PADRShadow, 1, CLOCK);


                  drp = BitRdPortI(PBDR ,  INPUT );

                  BitWrPortI(PADR, &PADRShadow, 0, CLOCK);
                  for(d2 = 0 ; d2< 1000 ; d2++)
                  {
                  }
    if(drp == 0)
  {
     droff = 0;

  }
  if(drp == 1 && droff == 0)
  {
       tcp_open(&socket,0 , host, 9090 , NULL  );
                  // Wait for the interface to come up
   d3=0;
  while (d3==0)
                  {
       d3 = sock_established( &socket );
                              tcp_tick(NULL);
       }

                  sprintf(buffer , "%02d", door);
```

```
        sock_write(&socket, buffer,2);

          sock_write(&socket,buffer,2);

                  sock_write(&socket,buffer,2);

                  sock_write(&socket,buffer,2);

    sock_close(&socket );

     droff = 1;
   }

     //*************************************************************
                   //  Refrigerator Sensor
    //*************************************************************
  BitWrPortI(PADR, &PADRShadow, 1, CLOCK);
                  rep = BitRdPortI(PBDR ,  INPUT );

                  BitWrPortI(PADR, &PADRShadow, 0, CLOCK);
                  for(d2 = 0 ; d2< 1000 ; d2++)
                  {
                  }
    if(rep == 0)
    {
     reoff = 0;

    }
    if(rep == 1 && reoff == 0)
    {
        tcp_open(&socket,0 , host, 9090 , NULL  );
                  // Wait for the interface to come up
     d3=0;
    while (d3==0)
                  {
        d3 = sock_established( &socket );
                        tcp_tick(NULL);
        }

                  sprintf(buffer , "%02d", ref);

        sock_write(&socket, buffer,2);

          sock_write(&socket,buffer,2);

                  sock_write(&socket,buffer,2);
```

```
                sock_write(&socket,buffer,2);

   sock_close(&socket );

    reoff = 1;
  }

  //*************************************************************
                  //  Water Sensor
   //*************************************************************
BitWrPortI(PADR, &PADRShadow, 1, CLOCK);
       for(d2 = 0 ; d2< 1000 ; d2++)
                {
                }

                mp = BitRdPortI(PBDR ,  INPUT );

                BitWrPortI(PADR, &PADRShadow, 0, CLOCK);

   if(mp == 0)
   {
     moff = 0;

   }

  if(wp == 1 && woff == 0)
  {
       tcp_open(&socket,0 , host, 9090 , NULL  );
                // Wait for the interface to come up
   d3=0;
  while (d3==0)
                {
       d3 = sock_established( &socket );
                        tcp_tick(NULL);
       }

                sprintf(buffer , "%d", motion);

       sock_write(&socket, buffer,2);

          sock_write(&socket,buffer,2);

                sock_write(&socket,buffer,2);

                sock_write(&socket,buffer,2);
```

```
      sock_close(&socket );

       moff = 2;
     }

   // force shift to Zero to load new data
         BitWrPortI(PADR, &PADRShadow, 0, SHIFT);
         for(d2 = 0 ; d2< 10000 ; d2++)
         {
         }

 }
}
```

4- PIC Code:

Serial Interface Code:

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        ToolTip1.SetToolTip(Me.Button1, "Start The Operation")
        ToolTip3.SetToolTip(Me.Button2, "Stop the Operation")
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        'Dim stro As String = 2
        Dim eV As Integer = 0
        Dim events(100) As String
        Dim ReceivedByte As Integer
        Dim I As Integer = 0
        Button1.Enabled = False
        Using com1 As IO.Ports.SerialPort =
My.Computer.Ports.OpenSerialPort("COM1")
            com1.StopBits = IO.Ports.StopBits.One
            com1.Parity = IO.Ports.Parity.None
            com1.DataBits = 8
            com1.BaudRate = 9600

            While STATE

                Dim d As Integer = 0

                ' Delay
                For d = 0 To 5000
                Next d

                Dim bits(8) As String

                If com1.BytesToRead > 0 Then


                    ReceivedByte = com1.ReadByte()

                    Dim msg As String = DecToBin(ReceivedByte)
                    If eV < 99 Then
                        events(eV) = msg
                        eV = eV + 1

                    ElseIf eV = 0 Then
                        events(eV) = msg
                        eV = eV + 1
                    End If

                    Dim j As Integer = 0

                    For j = 0 To 7
```

```
                    bits(j) = msg.Substring(j, 1)
                Next j


                'check for the read byte

                If bits(0) = 1 Then
                    sensor7.Checked = True
                End If
                If bits(1) = 1 Then
                    sensor6.Checked = True
                End If
                If bits(2) = 1 Then
                    sensor5.Checked = True
                End If
                If bits(3) = 1 Then
                    sensor4.Checked = True
                End If
                If bits(4) = 1 Then
                    sensor3.Checked = True
                End If
                If bits(5) = 1 Then
                    sensor2.Checked = True
                End If
                If bits(6) = 1 Then
                    sensor1.Checked = True
                End If
                If bits(7) = 1 Then
                    sensor0.Checked = True
                End If
            End If
        End While
    End Using

End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    STATE = False
    sensor0.Checked = False
    sensor1.Checked = False
    sensor2.Checked = False
    sensor3.Checked = False
    sensor4.Checked = False
    sensor5.Checked = False
    sensor6.Checked = False
    sensor7.Checked = False
    Button1.Enabled = True

End Sub
Private Function DecToBin(ByVal DeciValue As Long, Optional ByVal
NoOfBits As Integer = 8) _
As String

    Dim i As Integer
    'make sure there are enough bits to contain the number
    Do While DeciValue > (2 ^ NoOfBits) - 1
```

```
            NoOfBits = NoOfBits + 8
        Loop
        DecToBin = vbNullString
        'build the string
        For i = 0 To (NoOfBits - 1)
            DecToBin = CStr((DeciValue And 2 ^ i) / 2 ^ i) & DecToBin
        Next i
    End Function
    Private Sub Label2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub
End Class
```

Falling Sensor Code:

```
processor 16f628a
#include <P16F628a.INC>
__CONFIG _BODEN_OFF &_CP_OFF &_DATA_CP_OFF
&_INTOSC_OSC_NOCLKOUT &_LVP_OFF &_MCLRE_OFF &_PWRTE_OFF
&_WDT_OFF
org 0

;-------------------------------------------------------------------------------
Bank0 macro
        BCF 03h,5
        BCF 03h,6
endm
;-------------------------------------------------------------------------------
Bank1 macro
        BCF 03h,6
        BSF 03h,5
endm
;-------------------------------------------------------------------------------
Load macro Addres
        BCF 0Ch,4
        MOVF Addres,W
        MOVWF 19h
        NOP
        CALL Wait
        CALL Delay
endm
;-------------------------------------------------------------------------------
EnableAUSART macro              ;the sended byte is in 19h and 0Ch,4 is set after
complete sending
MOVLW 0x19
Bank1
```

```
MOVWF 99h               ;SPBRG is = 25 ---> baud rate is 2.4kB/s
BSF 98h,2               ;BRGH is = 0 (low speed)
BCF 8Ch,5               ;interrUpt is not desired
BCF 98h,4               ;enabling the Asyn. mode
BSF 98h,5               ;enabling the sending
Bank0
BSF 18h,7               ;enabling the serial port
BCF 18h,2               ;i am testing the no framing error
BCF 18h,1               ;also testing the overrun error
BCF 18h,6               ;Disable the 9th Bit
endm
;-------------------------------------------------------------------------------------------------------------
-------------------------------------------------
BSF 1Fh,2                       ;TURN THE COMPARATOR MODE OFF
BSF 1Fh,1
BSF 1Fh,0
MOVLW 0xFB
TRIS PORTB

EnableAUSART
CALL Delay
goto SEND
main:
MOVF PORTB,W
XORWF 20h,W
MOVWF 21h
BTFSC 21h,4
GOTO SEND
BTFSC 21h,5
GOTO SEND
GOTO main

SEND:
MOVF PORTB,W
ANDLW 0x30
MOVWF 20h

BCF 0Ch,4
movlw 0x00              ;       00H
MOVWF 19h
CALL Wait
CALL Delay
Load 20h

goto main
```

```
;=================================================================
=================
Delay:
X0:
DECFSZ 40h,F
GOTO X0
XX0:
DECFSZ 40h,F
GOTO XX0
RETURN

Wait:
A15:
BTFSS 0Ch,4
GOTO A15
RETURN

End
```

Medical Sensor Codes:

```
processor 16f628a
#include <P16F628a.INC>
__CONFIG _BODEN_OFF &_CP_OFF &_DATA_CP_OFF
&_INTOSC_OSC_NOCLKOUT &_LVP_OFF &_MCLRE_OFF &_PWRTE_OFF
&_WDT_OFF
org 0
;----------------------------------------
Bank0 macro
BCF 03h,7
BCF 03h,6
BCF 03h,5
endm
;----------------------------------------
Bank1 macro
BCF 03h,7
BCF 03h,6
BSF 03h,5
endm
;----------------------------------------
Load macro Addres
        BCF 0Ch,4
        MOVF Addres,W
        MOVWF 19h
```

```
        NOP
        CALL Wait
        CALL Delay
endm
;----------------------------------------
CheckZerosA macro AAA
MOVWF 30h
BTFSC 30h,0
GOTO AAA
BTFSC 30h,1
GOTO AAA
BTFSC 30h,2
GOTO AAA
BTFSC 30h,3
GOTO AAA
BTFSC 30h,4
GOTO AAA
BTFSC 30h,5
GOTO AAA
BTFSC 30h,6
GOTO AAA
BTFSC 30h,7
GOTO AAA

endm
;----------------------------------------
;----------------------------------------
CheckZerosB macro BBB
MOVWF 30h
BTFSC 30h,0
GOTO BBB

BTFSC 30h,3
GOTO BBB
BTFSC 30h,4
GOTO BBB
BTFSC 30h,5
GOTO BBB
BTFSC 30h,6
GOTO BBB
BTFSC 30h,7
GOTO BBB

endm
;----------------------------------------
```

```
EnableAUSART macro              ;the sended byte is in 19h and 0Ch,4 is set after
complete sending
MOVLW 0x19
Bank1
MOVWF 99h                       ;SPBRG is = 25 ---> baud rate is 2.4kB/s
BCF 98h,2                       ;BRGH is = 0 (low speed)
BCF 8Ch,5                       ;interrUpt is not desired
BCF 98h,4                       ;enabling the Asyn. mode
BSF 98h,5                       ;enabling the sending
Bank0
BSF 18h,7                       ;enabling the serial port
BCF 18h,2                       ;i am testing the no framing error
BCF 18h,1                       ;also testing the overrun error
BCF 18h,6                       ;Disable the 9th Bit
endm
;----------------------------------------
ReadPortA macro
MOVF PORTA,W
MOVWF 20h
endm
;----------------------------------------
ReadPortB macro
MOVF PORTB,W
ANDLW 0xF9
MOVWF 21h
endm
;----------------------------------------
BSF 1Fh,2                               ;TURN THE COMPARATOR MODE OFF
BSF 1Fh,1
BSF 1Fh,0
MOVLW 0xFF
TRIS PORTA                      ;11111111
MOVLW 0xFB
TRIS PORTB                      ;11111011

EnableAUSART
ReadPortA
ReadPortB

main:
MOVF PORTA,W
XORWF 20h,W
CheckZerosA SEND
MOVF PORTB,W
XORWF 21h,W
CheckZerosB SEND
```

```
GOTO main
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SEND:
ReadPortA
ReadPortB

BCF 0Ch,4
movlw 0x00
MOVWF 19h
CALL Wait
Load 20h
Load 21h
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

goto main

Delay:
X0:
DECFSZ 40h,F
GOTO X0
XX0:
DECFSZ 40h,F
GOTO XX0
RETURN

Wait:
A15:
BTFSS 0Ch,4
GOTO A15
RETURN

End
```

Stepper Motor Code:

```
processor 16f628a
#include <P16F628a.INC>
__CONFIG _BODEN_OFF &_CP_OFF &_DATA_CP_OFF &_LP_OSC &_LVP_OFF
&_MCLRE_OFF &_PWRTE_OFF &_WDT_OFF
org 0
;-------------------------------------------------------------------------------
Bank0 macro
        BCF 03h,5
        BCF 03h,6
endm
;-------------------------------------------------------------------------------
Bank1 macro
        BCF 03h,6
        BSF 03h,5
endm
;-------------------------------------------------------------------------------
Outvalue macro value
MOVLW value
MOVWF PORTB
endm
;-------------------------------------------------------------------------------
Chk60 macro Addres,OutAddres
        BTFSS Addres,3
        GOTO OutAddres
        BTFSS Addres,2
        GOTO OutAddres
        BTFSS Addres,1
        GOTO OutAddres
        BTFSS Addres,0
        GOTO OutAddres

        CLRF Addres
endm
;-------------------------------------------------------------------------------
EnableTimer0 macro
Bank1
BCF 81h,5
BCF 81h,3
BSF 81h,2
BSF 81h,1
BCF 81h,0
Bank0
endm
;-------------------------------------------------------------------------------
```

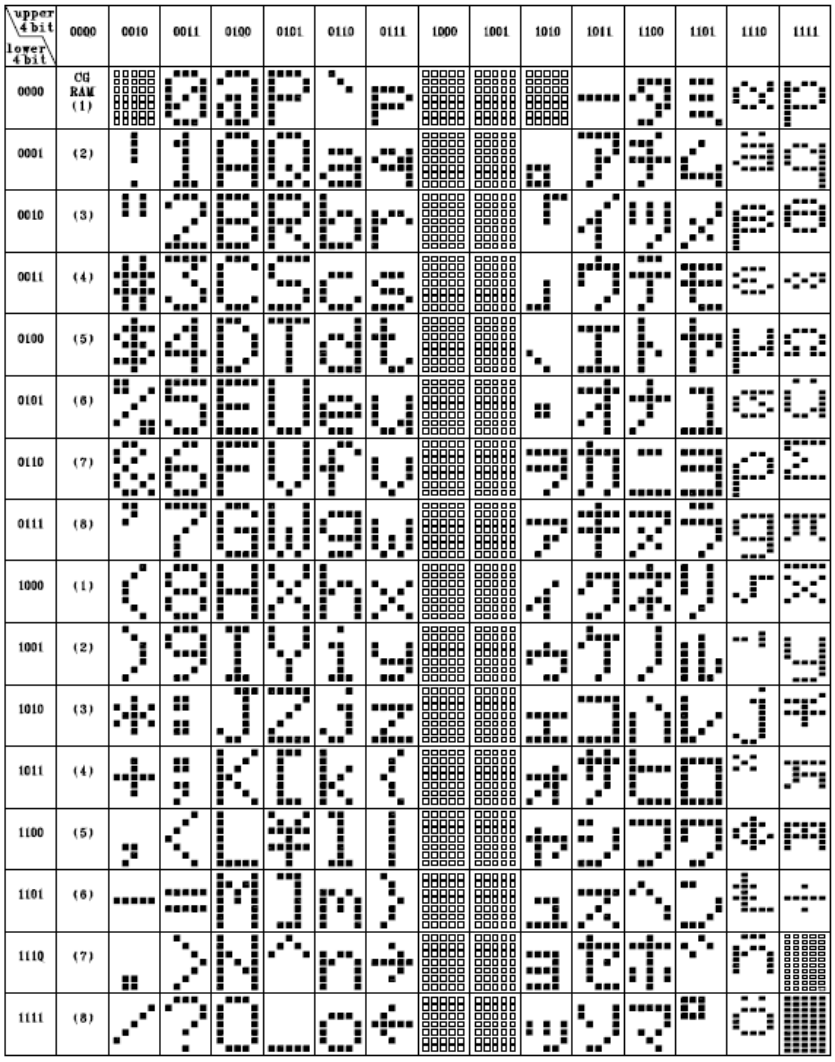
```
MOVLW 0x80                              ;1000 0000
TRIS PORTB
  MOVLW 0x01                            ;fbThe delay between the signals (effect
smoothness of the movement )
  MOVWF 23h
MOVLW 0xC9
MOVWF 24h
EnableTimer0
START:
BTFSC 0Bh,2
CALL TEMP
Chk60 30h,OUT
GOTO A0
OUT:
BTFSS PORTB,7
        GOTO START
A0:
BTFSC PORTB,7
        GOTO A0
MOVLW 0x04
MOVWF 25h
BCF 0Bh,2
CLRF 30h
;-------------------------          ;---------------------------------------------- |-------;
                                        ;1000                   |  1000
        ;
main:                                   ; This is one full step  |  1100
        ;
Outvalue 0x08                           ;0100                   |  0100
        ;
call Loop                               ;                       |  0110
        ;
Outvalue 0x0C                                   ;0010                   |  0010
        ;
call Loop                               ;                       |  0011
        ;
Outvalue 0x04                           ;0001                   |  0001
        ;
call Loop                               ; That is four full steps|  1001       This is eight half
steps    ;
                                        ;---------------------------------------------- |-------;

Outvalue 0x06
call Loop
Outvalue 0x02
call Loop
```

```
Outvalue 0x03
call Loop

Outvalue 0x01
call Loop
Outvalue 0x09
call Loop

decfsz 24h,F
goto main
MOVLW 0xC9
MOVWF 24h
DECFSZ 25h,F
GOTO main
GOTO START


Loop:
MOVF 23h,W
MOVWF 22h

add2:
DECFSZ 22h
  GOTO add2

return
TEMP:
INCF 30h,F
BCF 0Bh,2
RETURN
end
```

5- LCD Tables and Code:

Command	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0	Execution Time (f_{osc} = 250kHz)	Remark
DISPLAY CLEAR	L	L	L	L	L	L	L	L	L	H	1.64ms	
RETURN HOME	L	L	L	L	L	L	L	L	H	X	1.64ms	Cursor move to first digit
ENTRY MODE SET	L	L	L	L	L	L	L	H	I/D	SH	42µs	• I/D : Set cursor move direction I/D: H = Increase; L = Decrease • SH : Specifies shift of display SH: H = Display is shifted; L = Display is not shifted
DISPLAY ON/OFF	L	L	L	L	L	L	H	D	C	B	42µs	• Display D: H = Display on; L = Display off • Cursor C: H = Cursor on; L = Cursor off • Blinking B: H = Blinking on; L = Blinking off
SHIFT	L	L	L	L	L	H	S/C	R/L	X	X	42µs	S/C: H = Display shift; L = Cursor move R/L: H = Right shift; L = Left shift
SET FUNCTION	L	L	L	L	H	DL	N	F	X	X	42µs	DL: H = 8 bits interface; L = 4 bits interface N: H = 2 line display; L = 1 line display F: H = 5 X 10 dots; L = 5 X 7 dots
SET CG RAM ADDRESS	L	L	L	H	CG RAM address (corresponds to cursor address)						42µs	CG RAM Data is sent and received after this setting
SET DD RAM ADDRESS	L	L	H	DD RAM address							42µs	DD RAM Data is sent and received after this setting
READ BUSY FLAG & ADDRESS	L	H	BF	Address Counter used for both DD & CG RAM address							0µs	BF: H = Busy; L = Ready – Reads BF indication internal operating is being performed – Reads address counter contents
WRITE DATA	H	L	Write Data								46µs	Write data into DD or CG RAM
READ DATA	H	H	Read Data								46µs	Read data from DD or CG RAM

X : Don't care



INTERFACE PIN CONNECTIONS

Pin NO.	Symbol	Level	Description
1	VSS	0V	Ground
2	VDD	5.0V	Supply voltage for logic
3	VO	---	Input voltage for LCD
4	RS	H/L	H : Data, L : Instruction code
5	R/W	H/L	H : Read mode, L : Write mode
6	E	H, H → L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	NC	---	No Connection
16	NC	---	No Connection

We can use the port by using this program to send the data.

```
#define BYTE unsigned int
BYTE count;
void main()
	{
		if (count==0)
			{
			SendText("                ",16,2,0);
			SendText("No Medicine Used",16,2,0);
			SendText("                ",16,3,0);
			}
		else if (count==1)
			{
			SendText("                ",16,2,0);
			SendText("Use Medicine:",13,2,0);
			SendText("                ",16,3,0);
			SendText("        1",9,3,0);
			}
		else if (count==2)
		{
			SendText("                ",16,2,0);
			SendText("Use Medicine:",13,2,0);
			SendText("                ",16,3,0);
			SendText("        2",9,3,0);
			}
		else if (count==3)
		{
			SendText("                ",16,2,0);
			SendText("Use Medicine:",13,2,0);
			SendText("                ",16,3,0);
			SendText("        1 & 2",13,3,0);
			}
		else if (count==4)
			{
			SendText("                ",16,2,0);
			SendText("Use Medicine:",13,2,0);
			SendText("                ",16,3,0);
			SendText("        3",9,3,0);
			}
		else if (count==5)
			{
			SendText("                ",16,2,0);
			SendText("Use Medicine:",13,2,0);
			SendText("                ",16,3,0);
			SendText("        1 & 3",13,3,0);
			}
		else if (count==6)
			{
			SendText("                ",16,2,0);
			SendText("Use Medicine:",13,2,0);
			SendText("                ",16,3,0);
			SendText("        2 & 3",13,3,0);
			}
		else
			{
			SendText("                ",16,2,0);
			SendText("Use Medicine:",13,2,0);
			SendText("                ",16,3,0);
			SendText("      1 ,2 & 3",14,3,0);
			}
	}
	}
```