D'OH

DETAILED DESIGN REPORT

Table of Contents

Introduction

Purpose of the Document

The purpose of this document is to supply details of the design of our project. It is our goal to make any qualified colleague of ours to be able to implement this project after reading this documentation. All of the details necessary in order to start coding are included in this report.

Scope of the Document

This document consists of

- Project Description
- Project Goals and Objectives
- Constraints and Limitations
- Hardware and Software Requirements
- Project Diagrams
- Environments and Platforms
- Design and Implementation Schedules

Problem Definition

It may not be a subject relevant to all of us at young ages, but there is a certain problem that many people have to bear who have old/unhealthy relatives. Those people worry about their conditions. There are old people suffering from various diseases and the ones who are the concerned about their situation are not always eligible to take care of them. Maybe they have a job with not flexible working hours, they live in a different city, or they are occupied with something else.

However, these people may really need good care. Their health values may change suddenly, they may fall etc. there should be someone to keep things under control. It may be difficult to find someone to nurse these patients; also people may have difficulties relying on them.

Instead of this we created our own solution, by this way people can be informed about the situation of their relative's health and work on other things they have to, at the same time.

Group Members and Roles

We can name Buğra as our group leader because he is the one who shares the workload among the group members. He usually studies to that part of the project and attends our meetings with some pre-knowledge. He sets a plausible deadline for every task which has to be accomplished and that way we had ease managing with this project.

Güven always wants to be sure of the plan, that it will work, before applying. He likes keeping his skeptic attitude towards new ideas; hence they would not lead to time losses. After his objections are concluded, he briefly summarizes the situation as the summarizer of the group.

The most optimistic of all the group members is Özkan. He always has higher hope about the project than any of us, which helps also raising the motivation of others. Also he is full of innovative and creative ideas, he can come up with a totally different suggestion, let us cope with the problems better.

When there is the need of doing research Gizem is the obvious choice, she is incredibly talented using research sources. She composes an archive of things she has researched with the most essential and useful documents. She always tries to record anything about the project; its situation, things that we do etc.

Overall Description of the Project

As mentioned earlier, the project has three main parts. In the first part, we deal with the sensors and their connection with the pic board. The second part's main concern is the mobile applications and data transfer via Bluetooth between the source device containing the sensors and the receiver device (a mobile phone) which would publish the data. And this

brings us to the third and the final part which is based on this action of publishing data over the web for the ones interested with this data.

Phase One: Embed Orchestra

In the first part of the project, most of the time, we would be dealing with maintaining the connection and obtaining the flow of data between the output ports of the different sensors that we would use and the microcontroller that has a Bluetooth transceiver on it.

To be able to work on data the sensors give, without no doubt, we would first need to learn the way which each and every one of the sensors output the data. After having the connection between the output ports of the sensors and the analog or digital ports of the microcontroller, we would pack the data collected over the ports and push them to the Bluetooth device connected.

Phase Two: Mobydick

In this second and the most challenging part of the project, we dived into a world that we were not familiar with. Mobile Applications were always out of our scope until this project. Our purpose is to receive sensor data over Bluetooth with the use of a mobile application installed to a java supporting cell phone, and send the packaged data to the web via GPRS. The mobile application is an intermediate station through the data transformation from the sensors to the web. It has a very simple user interface since it works In the background and no continuous interaction with the user is necessary. The menu will let the authorized user to change the options. The authorization is needed to protect the data flow from attacks or accidental moves. Additional security measurements are added to ensure the reliability of our software. The application establishes a Bluetooth connection immediately when it is started and the connection will stand still during the runtime to get sensor data without any loss. The other job of this application is to send the periodically collected sensor data to a web server over GPRS. GPRS connection will also be established using a predefined socket

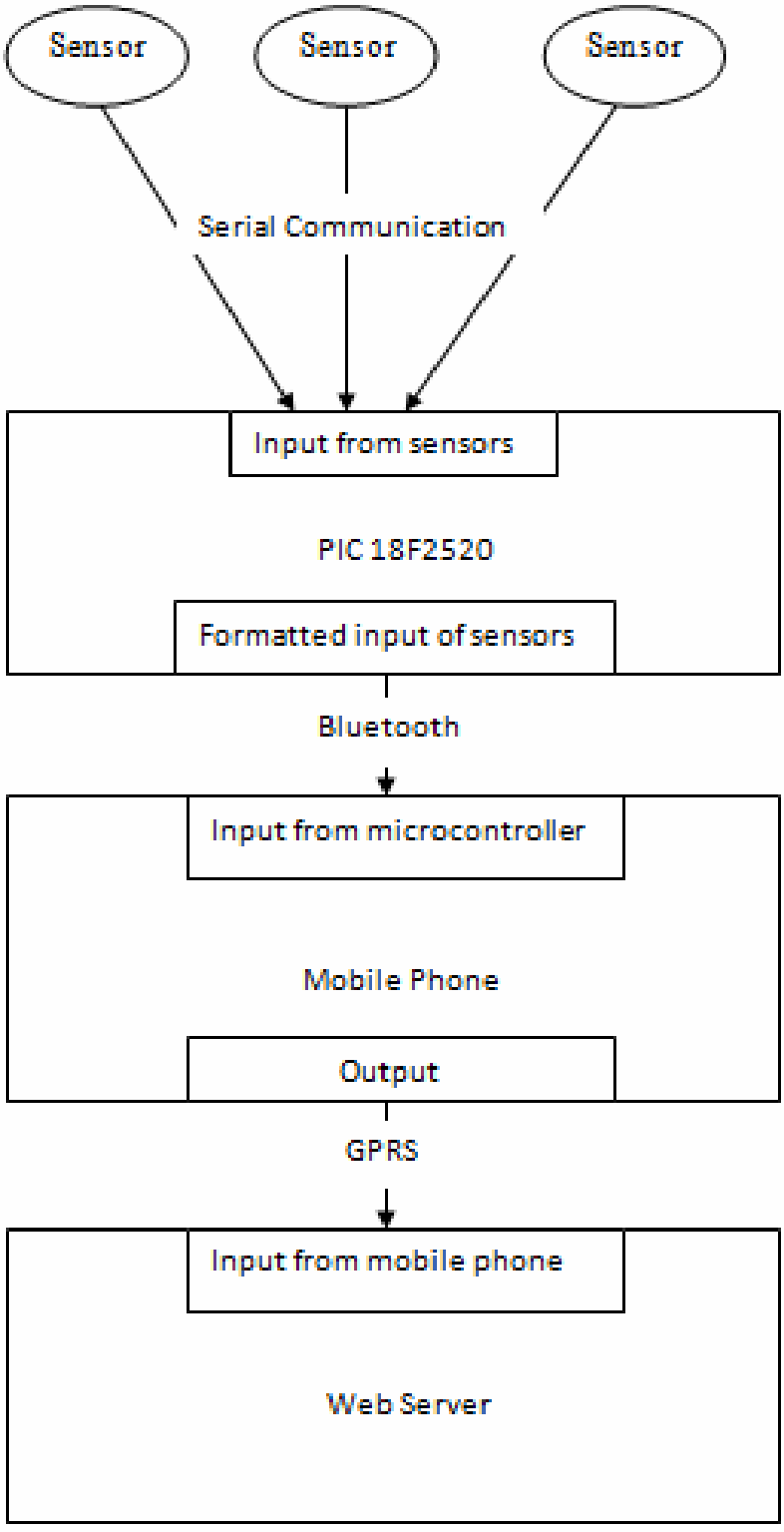
with the remote server as soon as the application runs. So the mobile application will serve as a secure bridge for the data flow from the sensors to the web.

Phase Three: SpiderPig

This phase of the project would most probably be the most straightforward one among all, but nonetheless quite essential. It will fulfill the need of complete interface in our compact system. The web server is going to provide users a quick, efficient way of monitoring the latest condition of the patient. The data collected over GPRS connection, will be published through a webpage. Data from the mobile application of the patients will be gathered into a large pool. To handle this vast amount of data we are going to employ a database management system in our web server. By this way, old records will also be accessible while the newest ones will be the first to be displayed in the web page of the patient. The web page will be designed in such a way that while the doctors have access to all of the details about the condition of the patient, the patient's relatives will be informed about the overall situation of the patient without letting them be lost in details.

Architectural Overview

If we look at the system as a whole it is easy to observe that, it is distributed, and consists of different platforms to develop software. This makes the partitioning of the system easier, but brings the difficulty to specialize in multiple programming paradigms. The abstract architecture of the whole system can be seen in the below figure.


Sensor
Sensor
Sensor
Serial Communication
Input from sensors
PIC 18F2520
Formatted input of sensors
Bluetooth
Input from microcontroller
Mobile Phone
Output
GPRS
Input from mobile phone
Web Server

As can be seen from the figure also, there is not a center point for the whole system where everything is managed. In fact the system can be viewed as a series of data transfers starting from the sensors and ending in the web server.

Data flow is the primary process, thus in the figure which is the abstract system only data transfer parts are shown. There are additional parts employed in each of the points where data passes through.

Three different programming platforms will be used to achieve the final product, first is PIC programming, second is mobile application programming, and the last one is web programming. Platforms are quite diverged from each other.

Microcontroller programming part consists of three major parts. In the first part where measurements of the sensors are sent, there should be a robust PIC program to establish a reliable serial connection between the sensors and microcontroller. After that received data should be compressed in an efficient way to be ready to send. In the last part data is put into the ports of microcontroller hence Bluetooth device can read and send it to the mobile phone. All of these tasks will be coded in C; MPLab, the official IDE of the Microchip, provides this feature.

Mobile programming part is the first part where received data is evaluated. Because in the previous parts data was only received, packaged and sent. Our mobile program will check the values in the data package and will run script codes to send notifications to the doctor and relatives of the patient. We'll use built-in methods of J2ME to supply this. But mobile program will contain more vital parts than this. It should receive data from the Bluetooth device and send it readily to web server via GPRS.

Web programming part includes also database programming, otherwise it'd be impossible to monitor the conditions of each patient for such a big system. Data stored in the web server will be combined with a user-friendly web interface to provide users (doctors, patients, and their relatives) an easy way to monitor the latest conditions.

Phase 1: Embed Orchestra

The sensors will periodically produce outputs which are going to be collected and checked by the PIC program and then the packed data will be transmitted to the Bluetooth device connected. Throughout this flow of data from the sensors to the BT device, there will be pauses and triggering processes caused by alarm conditions. One of the most important alarm conditions is occurred when the critical sensors output a value outside of the predetermined range. Another one will happen when the patient hits the button integrated on the module having the sensors, PIC and BT device. One other example may be the situation when one or more of the sensors run out of power and stop giving outputs. In an alarm condition, PIC will interrupt the process and send an appropriate alarm code via BT immediately.

During the normal flow of the data there are some points to focus on. Firstly, the format of the sensors data should be standardized. Each sensor may have a different format for outputs and even different output types. One may need an analog pin while the other one requires digital. The PIC program that we will develop will handle all of the possible conditions and transform these unformatted inputs taken from the sensors to a standardized form and then pack them. We need to be more specific about the sensor output types and formats at this point. However, thanks to the company that we had contacted for information about an all-in-one device containing some of the sensors that we would include in our project, we have no further information for now. But we are working on a way of gathering the necessary information without the sensors. We are planning to emulate the sensors in a realistic manner. Another point is the size of the input and the output of PIC. The size should be adjusted for the optimal progressing of the device. By this, we mean the output transmitted to the BT device should be packed so that the size of the packet should not exceed a threshold value but should contain enough data. The frequency of transmitting data and the average time cost of one transfer over BT should be calculated to obtain the cost. These are our major considerations for the embedded module.

Module is a small device which patients wear to their wrist. Device is comfortable, fashionable and be used like an accessory. Bluetooth, microcontroller, battery, cables and

the cover are the main parts of module and the total weight of module is smaller than 100 gr. Actually it is like a heart rate monitor. Below is a picture of a heart rate monitor.

The main part of the module work with a CR2032 battery. Sensors have their own batteries in this way we save batteries. Module has two buttons : turn on/off button, emergency button. Both of these buttons have crucial importance in the project. Turn on/off button is to save energy while the module is not being used. It is a small button thus user, the patient, avoid pressing it by mistake. Patients press emergency button in emergency situation. Emergency button is the quickest way the patient can want help, pressing this button will send notifications to relative, doctor and call the emergency service. Thus patient should use it under serious conditions. Module does not have a monitor. This module is being worn by old people and will not have a direct electricity connection so power consumption is a critical issue. Moreover there will be cell phone close to the patient (to establish the bluetooth connection) where the patient can track down these values.

There will be 5 sensors included in the module:

-Blood Pressure

-Pulse Pressure

-Body Temperature

-Heart Rate

-Oxygene Saturation

We are using these 5 sensors because the values of these sensors measurements are the most vital, and also the most unstable health values. They can vary in such small time periods. The other values which can be utilized instead of the ones we have selected are either less essential or more stable thus don't need to be measured in small periods.

Normal values:

-Blood pressure : 115 mmHg diastolic, 75 mmHg systolic

-Pulse Pressure : 40 mmHg

-Body Temperature : 36.8 °C ±0.7 °C

-Heart Rate : 130 - 150 bpm

-Oxygen Saturation : 90%

Critical values:

-Blood pressure: diastolic over 180 / systolic over 110

-Body temperature: Dying values are 28 (low) - 42 (high).

Oxygen saturation: 50%

Data types of the sensor values are short, the units of these values will not be transferred from any device to another one but while displaying them the units will be added.

-Blood pressure : mmHg

-Pulse Pressure : mmHg

-Body Temperature : °C

-Heart Rate : bpm

Presently we don’t own any of these sensors, though we researched sensors in the internet widely. There are sensors in market but they are designed for final user so we cannot embed them into our module. Most of them have their own displays and are not suitable for connecting to our module. Beside this, these sensors are really expensive and which brings financial considerations.

Simulating sensors instead of using real ones has some disadvantages. We are planning to build a complete system but without sensors data we carry will be artificial. At the end of the project we won't know how the system will work with sensors in real life, which is a major issue about our system. On the other hand simulating sensors has some advantages. There may be problems in the connections or running of sensors with the embedded module and just for this reason we would not be able to focus on our project which will hinder our progress.

Simulating sensors is very important issue. Algorithms must output random results as close as the values of real sensors. Thus having a well-thought, reliable randomization algorithm is a critical issue. In most programming languages built-in randomize functions make use of CPU clock time for the seed value. This seed value is multiplied,summed and modulated with constant values to achieve nearly random values. We are going to replace CPU clock time with the values of timers located in the microcontroller, while assigning the seed value. Below are our procedures to obtain random values for each 5 sensors:

$blood\_pressure_{k+1}$

$= blood\_pressure_k +$

$(timer\_value\_1 * blood\_pressure_k + const\_1) \bmod const\_6 -$

$const\_6/2;$

$pulse\_pressure_{k+1}$

$= pulse\_pressure_k +$

(timer_value_2 * $pulse\_pressure_k$ + const_2)mod const_7 -

const_7/2;

$body\_temperature_{k+1}$

= $body\_temperature_k$ +

(timer_value_3 * $body\_temperature_k$ + const_3)mod const_8 -

const_8/2;

$heart\_rate_{k+1}$

= $heart\_rate_k$ +

(timer_value_4 * $heart\_rate_k$ + const_4)mod const_9 -

const_9/2;

$oxygen\_saturation_{k+1}$

= $oxygen\_saturation_k$ +

(timer_value_5* $oxygen\_saturation_k$ + const_5)mod const_10 -

const_10/2;

Randomization functions are close to the common ones. The extra subtraction operand in the end makes the possibility of resulting values to decrease with 50 % chance. Thus values are closer to the real sensor values and if we assume the module operand is unbiased, random values will converge to a value. (Because the possibility is the same whether the obtained value is bigger or smaller than the previous one)

We are creating packages while publishing data to the mobile phone. Mobile application can not interpret data if we do not use a package. In addition to this there can be data loss we are not aware of in lack of packages. Any small data loss is not acceptable. We use “Do'h data transfer protocol” to create packages. Do’h data transfer package consist of:

starting byte

package ID

data

checksum

Starting byte is to show that package belongs to our protocol. It is 0xd0. If a package does not start with this ID our mobile application will not handle it. There two different type of packages: ordinary situation package , emergency situation package. Package ID is to understand which type of package it is. ID of emergency package is 0x0e and ID of ordinary package is 0x08. Data of emergency package show the type of emergency.

0x00 -emergency button pressed

0x01 - Blood pressure value is too high/low

0x02 - Pulse Pressure value is too high/low

0x03 - Body temperature value is too high/low

0x04 - Heart rate value is too high/low

0x05 - Oxygen saturation is critical

Data of ordinary package consists of 5 sensor data. Each sensor data is 2 bytes long . There is only one data for each sensor. If sensor frequency is higher and we get too much data from that sensor ,we take avarege of that data and add to the package. We know that if data is below or above the normal gap we create emergency packages. If the data is in normal gap we take the average so we don't need to send it more than one. First two bytes are blood pressure data,next two bytes are pulse pressure data, next two bytes are body temperature data, next two bytes are heart rate data and last two bytes are oxygen saturation data. And finally we have a checksum byte. A checksum is a fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting accidental errors that may have been introduced during its transmissions. We calculate checksum with this algorithm.

```
Byte *bt  // our data

int length // lenght of our data

Byte b -> bt[0]

for(int i=1;i<length;i++)

        b^=bt[i]
```

b is our checksum byte. We are sending all sensor data in one package. We could create different packages for different sensors but to save batteries we send all data in one package. Saving means sending data with bluetooth consumes too much energy. If we create different packages for different sensors that will be meaningless.

We are sending data in every two minutes. Period is very important in our project. Some data has to be send very fast but if the period is too long that causes problem. Beside this if we send too many data that causes too many bluetooth connection and this means too much energy consumption and too much data can be unnecessary. So we have to choose period very carefully .It must not be too short or too long.

Our module has an emergency button on it. Patients press it when they do not feel good or something has happened like they may has fallen dawn or they may hurt theirselves. When patient press that button emergency package is going to be created immediately.If the module is sending a package ,it stops sending that package and sends emergency package. This button is to save more lifes.

Part Two: Mobydick

The mobile application part, Mobydick, also consists of three parts which are Bluetooth connection, internal processes and GPRS connection parts. To help the reader have a better

insight, we chose to describe the design by splitting the hardware related part and software related part.

The software development is the center of this part contrary to the hardware oriented first part.

We decided to use Java in the implementation of the mobile part. The main reason why we have chosen Java language is the portability it offers. Most of today's cell phones have Java support and that will let our mobile software run on almost all of the mobile phones supporting java. It also gives us the ability to design the user interface easily. However, Java's functionality varies by headset used. What we are capable of doing mostly depends on the JSR's included in the phone. For some of the headsets, it does not support high resolution pictures and has limited file access. But these are not big problem, considering that our mobile software will just serve as a bridge in the flow of data and it will not use a complex user interface with high resolution pictures and the limited file access will be more than enough for the job it is doing.

Mobydick, consists of three subparts namely *the easy-come, the briDge* and *the easy-go* which are explained in detail here.

The **easy-Come** part is where we take care of the Bluetooth connection and gather the data packages over BT.
The **briDge** part is where we have our user interface and handle internal processes.
The **easy-Go** part is where we establish the GPRS connection and send the data (or error reports) over internet.

1) The easy-Come

Having a connection with the remote module over Bluetooth is the first step to be taken. Bluetooth connection requires both devices to be active. Also, pairing between them is needed for the communication to be started. The remote module is permanently publishing

the data packages and only goal of ours is to catch this continuous flow and pass it to "the briDge".

To make use of Bluetooth connection in a Java-coded mobile application, JSR-82 is the only present choice. JSR-82 is an interface where APIs that allow Java middlets to use Bluetooth connection, are described. The problem with JSR-82 is that, not all the Bluetooth compatible devices support it. It lessens the number of available phones to a few high-quality ones, most of the mobile phones have hardware built in but still does not support this technology. The list of available ones can be viewed in this link:

http://code.google.com/p/bluecove/wiki/phones .

Some of the capabilities of JSR-82:

- Local Bluetooth device settings management.
- Neighborhood Bluetooth device discovery.
- Bluetooth device search on the already discovered Bluetooth devices.
- Communication connections control and management.
- Security to all of the options above.

Some of the Local Device Methods of JSR-82

public static LocalDevice getLocalDevice()

This method returns an instance of our local device (cell phone). Multiple calls to this method will return the same object since we have a single local device.

public java.lang.String getBluetoothAddress()

This method returns the 6 byte long unique Bluetooth address of the local device.
Ex: 00:23:1d:6c:80:e6

public DeviceClass getDeviceClass()

This method returns the class of the local Bluetooth device as an object of DeviceClass. The class contains information about the device type and the services it supports.

public boolean setDiscoverable(int mode)

The method is used to make the local device discoverable or not. The predefined integer values are DiscoveryAgent.GIAC, DiscoveryAgent.LIAC, DiscoveryAgent.NOT_DISCOVERABLE.

public int getDiscoverable()

This method returns the current mode of the bluetooth device.

public DiscoveryAgent getDiscoveryAgent()

DiscoveryAgent class exports methods to search for both Bluetooth devices and services. A local device can have only one instance of discovery agent and this is retrieved using the above method.

The important point is that since it is of vital importance to make this system working correctly, it should be a secure connection so that no interruptions or intrusions can occur. Loss of data is not acceptable. So besides the powerful features of this API such as a complete management of the local device and easy search, connection and communication offered, the security JSR-82 provides, is the key point for this part of our system.

2) The briDge

The briDge, as the name implies, serves as an intermediate stage in the data flow. The information gathered from "the easy-Come" will be passed to "the easy-Go". Apart from its main job, the briDge also has lots of features such as a graphical user interface including an options menu, the ability to send SMSs and make phone calls when necessary.

To start with its main duty, the briDge, getting data packages over the bluetooth connection, immediately does a verification of the data's completeness by the use of checksum. If a package is verified, it is saved into a temporary file for short term archiving and if not, an error report including an explanation and the checksum result is prepared and saved into the file instead.

The data file format is kept as the way it is for the sake of simplicity. Each sensor data is written in a different line. As additional items, the date and time information will also be stored in these files. The file format will be:

The sensor #1 data\n
The sensor #2 data\n
The sensor #3 data\n
The sensor #4 data\n
The sensor #5 data\n
MM/DD/YYYY\n
hh:mm:ss

The error report will include an explanation of the report which can be edited in the first line. Then the checksum value received and the checksum value calculated in the second and the third lines respectively. The date and time information is also added to this report as the last two lines.
The format is illustrated as follows:

“This is the explanation”\n
ChecksumReceived\n
ChecksumCalculated\n
MM/DD/YYYY\n
hh:mm:ss

This is the situation when everything goes just fine. However, in an emergency condition the system will behave differently to be able to handle the emergency correctly and rapidly. An

emergency situation occurs when a value obtained from one sensor is very close to the borders of the vital range or the emergency button is pressed.

In case of an emergency, the application immediately sends SMSs to the predefined receivers and calls the emergency number.

The mobile application will create two different emergency messages and the SMS receivers will be separated to two groups as the doctor(s) and the patient's relatives to be able to send the meaningful data report to the relevant receiver. The message prepared to be sent to the doctor will include the last five measurements and time of the alarm. The other type of message will only contain a predefined message for this type of alarm condition to notify the patient's relatives about the situation.

SMSs will be sent over GPRS using one of the hundreds of SMS service providers. By this way, the application will not depend on the GSM company's SMS service which is used in the phone. These service providers offer fast and cheap text data transformation, and so there will be no loss of time when a stable and known service provider is selected. The cost of this SMS sending process is negligible.

The application will also call the emergency number (112) and play a record which gives information about the patient's name, address and blood type and ask for immediate help. It will repeat calling 112 and sending messages periodically until an authorized user shuts down the alarm by logging in to the application.

The User Interface

The user interface will be designed in a simple manner aiming the ease-of-use even it would not be used frequently. When the application is started, a log-in form will appear. The log-in mechanism will be used for hindering the access of anyone but the doctor or the family members. The authorization is needed to protect the data flow from physical attacks or accidental moves.

Figure – 1 Figure-2

The log-in system will make a simple username-password checking to be able to decide whether to give the user authorization or not. A list of user names and passwords will be kept in an encrypted file in the cell phone memory. When the “submit” button is pressed, the application will get the values from the “user name” and “password” fields and try to find a matching in the file. If there is a matching, it will let the user in. When the password or the user name is entered incorrectly, it will be asked for 4 more times for the correct log-in info. After 5 attempts to log-in, the system will decide that an unauthorized person is trying to log-in and so lock the log-in mechanism for 10 minutes.

Once logged in, the user will see a menu including the following options:

- Change password
- Change SMS Receivers

- Report Options
- Connection Options

Figure-3

Change Password

The user will be able to change his/her password by entering the old password once and the new password twice in the fields provided. Unsuccessful attempts will not make any effect.

Change SMS Receivers

The mobile phone numbers which are selected as SMS receivers can be changed from here. The receivers are categorized as doctors and others as mentioned before.

There will be up to ten phone numbers to be stored.

Report Options

This menu item will lead to a sub menu with the following options:

- Edit Error Report
- Edit Emergency Report

Edit Error Report

Partially change the content of the report which is delivered when an error occurs. Saying partially, we mean that some parts of the error message (including the error code) can not be edited.

Edit Emergency Report

Change the content of the emergency report which is sent to the receiver(s) other than the doctor(s).

Connection Options

This menu item will lead to a sub menu with the following options:

- Change SMS Service Provider
- Change GPRS Service Provider
- Change Remote Bluetooth Module Address
- Change Web Server Address
- Change Transfer Frequency

Change SMS Service Provider

Select an SMS Service Provider from the list

Change GPRS Service Provider

Select the active GPRS Service Provider from the list

Change Remote Bluetooth Module Address

Enter an address for the Remote Bluetooth Module in the hexadecimal format.
Ex: 00:23:1d:6c:80:e6

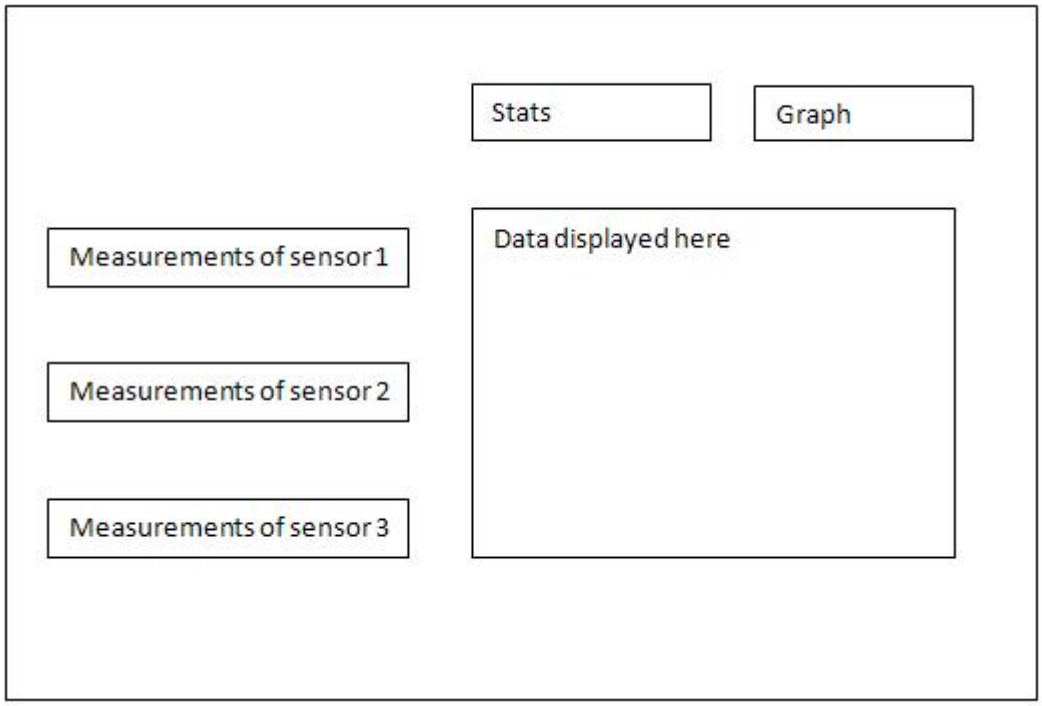
Change Web Server Address

Change the address of the Web Server to where the data are sent

Change Transfer Frequency

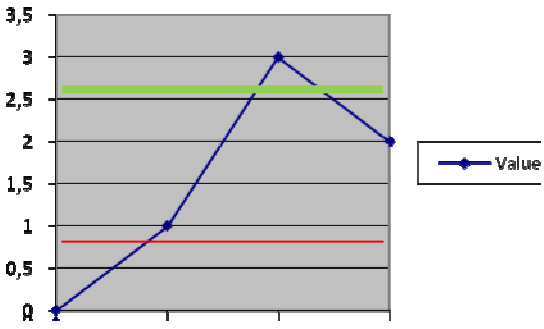
Change the frequency of data transfer from the mobile phone to the web server (in seconds)

Other than these operations, which an authorized user can perform with the mobile application, there is also another important feature displaying the received data. A generic representation of this interface is in the below figure.



The user can select in which way he/she is going to view the data from the upper part, when stats is selected measurements are displayed line by line in time order. Values bigger or smaller than threshold values are differently colored.

In the graph mode the values will be displayed in a graph, there will be two lines to indicate min and max norms. In this symbolic graph below, green line denotes the max norm value and the red line denotes the min norm value.



3) The easy-Go

The final part of the Mobydick is where we establish a GPRS connection to send the sensor data obtained to a web server which we will store them permanently.

The GPRS connection is very straightforward. A client – server relation will be produced between the remote web server and the cell phone over a socket connection.

A socket will be opened in the mobile phone like:

SocketConnection sc = (SocketConnection) Connector.open("socket://"+ theAddressAndPort);

Then, using an output stream data will be sent over GPRS:

OutputStream os = sc.openOutputStream();
os.write("Connection is established!");

The datum will be sent to the web server with the frequency set from the options menu. It is not allowed to ask for a frequency higher than the maximum value possible. The maximum frequency is determined by the speed of GPRS connection.

Constraints and Possible Problems

1) The Battery Life

The Bluetooth connection, as may be guessed, will affect the stand by duration of the cell phone. Consuming more energy, battery will be exhausted in a shorter period of time when the Bluetooth is on. However, this is not leading to a problem since the cell phone will not be used on the run. Instead, most of the time, it will be on a table or somewhere at the center

of the house to be able to make the best of the range of Bluetooth in the house's internal area. Therefore, the cell phone will easily be connected to a charger unit when the battery is low and so no interruption in the flow of data will occur.

2) Daily Use of the Mobile Phone

The design of the mobile application is based on the assumption that the cell phone it is installed on will not be used for another purpose such as daily use. The reason why we have assumed it is that the java application we will design, once started, should be active and any other application or service should not interrupt the process. The incoming or outgoing calls, SMSs, other programs etc. will pause or maybe stop our program which is not acceptable. The development of a run-in-background program is a solution but most of the cell phones supporting java programs unfortunately do not support Java Midlets to be run in the background so this choice will cause the loss of portability and we definitely do not want this. In our trials, we saw that the incoming calls are only pausing the application until the end of the call but outgoing calls cannot be made before closing the application. Because of these reasons, we prefer to assign a cell phone on which patient will not receive and make calls to be used in our project.

Environment

Netbeans IDE and Java ME SDK are used for Mobile Application Development. Both tools have cell phone emulators and sample projects. The platforms' ease of use and number of useful features for developers have been effective on our choice of the environment. Detailed information for the tools can be obtained from:

- *http://www.netbeans.org/features/javame/index.html*
- *http://java.sun.com/javame/index.jsp*

Phase 3: SpiderPig

Data obtained from the web server will be collected in a carefully designed database. Each new package of information will be unpacked and placed in the tables of the Database using MsSQL. The newest entries will be fetched periodically and displayed on the web site. Over the site, old records for the patient will be reachable for better diagnosis and controls. The records can be viewed hourly, daily, weekly and monthly and there will be charts showing the changes in the values. There will be a log-in mechanism in the site for hindering the access of anyone but the doctor or the family members. There will be different perspectives for these two different log-in types. The doctor log-in will lead to doctor perspective and normal log-in will enter the site for the family. The main difference between these two perspectives is the amount and the type of information displayed. In doctor perspective, doctor will have the opportunity to check each step of the patient with all the measurements, warnings etc. Doctor will have the opportunity to add notes, configure the charts as he/she wants and access to all the information in the database. Doctor may also edit the preferences such as frequency of arriving data up to a certain limit (sensors put this limit). Normal log-in, on the other hand, will not cause the users to be choked in details that would mean nothing to a person (if he/she is not a doctor). The overall condition of the patient will displayed. But in case they need more (less) than that, users will have the opportunity to increase (decrease) the level of details up to a point. These are the expected features of the web server explained without any technical detail.

REQUIRED PROGRAMS

For this part of the project, some development tools and utilities are used. Before we started our project, we tried to understand which programs we will need and after deciding that, we started to install the required programs to our computers.
In the web designing part of our project, we used Visual Studio 2008, IIS 7.0 and MsSql 2005.

IIS, Internet Information Services, is a set of internet-based services. We decided to use this because we are using ASP.NET and Visual Studio. Because of being all of them are Microsoft company products, we preferred IIS for the sake of compatibility.

We used Visual Studio as a platform in the web part of our project. Microsoft Visual Studio is an Integrated Development Environment (IDE). It can be used to develop console and graphical user interface applications along with windows forms applications, web sites, web applications, and web services.

We chose Visual Studio as our design platform because of our choice of ASP.NET and C#, again. All of them are developed by Microsoft and even thinking them separate is nearly impossible.

WHY ASP.NET and C# ??

We used ASP.NET and C# in web design part of our project. ASP.NET is a web application framework developed by Microsoft, and C# is an object-oriented and functional programming language developed by Microsoft.

The advantages of ASP.NET:

- improved performance
- high qualified tool support
- power and flexibility
- simplicity
- controllability
- scalability
- renewability
- compiled code
- drag & drop feature
- unrelated data access
- session state
- security support
- hosting choices
- more quick web applications

- powerful database functions
- being a part of .net
- separate pages for program and HTML

Moreover, we could use ASP.NET with a different programming language besides C# but we preferred it because we were familiar with C-originated languages and C# is the best possible choice for us because of the mentioned compatibility reasons.

We used MsSQL as database server, because we are making the web part Microsoft based and we can establish the connection between the web page and the database easier with MsSQL rather than the others like MySQL or postgreSQL.

DATABASE DESIGN

There is 6 database tables in the web part of our project:

- Doctor Table
- Relative Table
- Patient Table
- Emergency Table
- Message Table
- Measurements Table

1. **Doctor Table:**

The fields of the doctor table are : user name, password, name, surname, tel., address, mail , id (doctor id)

Column Name	Data Type	Allow Nulls
id	int	☐
name	varchar(50)	☐
surname	varchar(50)	☐
username	varchar(50)	☐
password	varchar(50)	☐
tel	varchar(20)	☐
address	varchar(200)	☐
mail	varchar(50)	☐
		☐

The screenshot of our doctor table. The primary key is id.

2. **Relative Table :**

The fields of the relative table are : user name, password, name, surname, tel., address, mail, patientid, relativeid

relativeid	int	☐
name	varchar(50)	☐
surname	varchar(50)	☐
username	varchar(50)	☐
password	varchar(50)	☐
tel	varchar(20)	☐
address	varchar(200)	☐
mail	varchar(50)	☐
patientid	int	☐
		☐

The screenshot of our relative table. The primary key is relativeid.

3. **Patient Table :**

The fields of the patient table are : name, surname, patientid, notes, doctorid, age

	Column Name	Data Type	Allow Nulls
🔑	patientid	int	☐
	name	varchar(50)	☐
	surname	varchar(50)	☐
	date	datetime	☐
	notes	varchar(1000)	☐
	doctorid	int	☐
▶	age	int	☐
			☐

The screenshot of our patient table. The primary key is patientid.

4. Emergency Table :

The fields of the emergency table are : patientid, emergencyid, emergencyValue, date

	Column Name	Data Type	Allow Nulls
▶	patientid	int	☐
🔑	emergencyid	int	☐
	date	datetime	☐
	emergencyValue	float	☐
			☐

The screenshot of our emergency table. The primary key is emergencyid.

5. Message Table :

The fields of the message table are : messageid, message, doctorid,patientid, readUn, direction

	Column Name	Data Type	Allow Nulls
🔑	messageid	int	☐
	message	varchar(1000)	☐
	doctorid	int	☐
	patientid	int	☐
	readUn	bit	☐
	direction	varchar(50)	☐
			☐

The screenshot of our message table. The primary key is messageid.

6. **Measurements Table :**

The fields of the measurements table are : patientid, measurementsid, measType, measValue, date

	Column Name	Data Type	Allow Nulls
	patientid	int	☐
🔑	measurementid	int	☐
	measType	varchar(50)	☐
	measValue	float	☐
▶	date	datetime	☐
			☐

The screenshot of our measurements table. The primary key is measurementid.

FEATURES OF THE WEB PAGE

1. **Different Perspectives for Doctor and Relative**

There are two different perspectives:

- Doctor’s Perspective
- Relative’s Perspective

There are some differences between the doctor's perspective and the relative's perspective.

In relative's perspective, the user can only reach his patient's health values. However, in doctor's perspective, the user (the doctor) has different patients and he may select any of them. So, there should be a field that the doctor can write the name of the patient and reach his values.

Moreover, in relative's perspective, there is nothing different from the table of values but the doctor should get some notes about the patient and so there should be notebook for each patient. With this, the doctor can note his thoughts about the patient.

2. Pages of the Web Site

There are 9 different pages of the web site:

1. Doctor/Relative Login:
This page is an ordinary login page (consists of text boxes to write user name and password)

2. Doctor Homepage:
Doctor home page includes a search field to search among all of the patients, messages button to write and read messages and the emergency field for his patients’ emergency situations.

3. Relative Homepage:
Relative homepage consists of his/her patients’ health values, message button, and emergency field.

4. Doctor Patient Search:
From this page, the doctor would make a detailed search for his/her patients.

5. Doctor Show Patient:

In this page, the doctor reaches the values of the patient that he searched. (Search Results)

6. Doctor Show Message:

The page for doctor's messages.

7. Doctor Write Message:

The page for doctor to compose messages.

8. Relative Show Message:

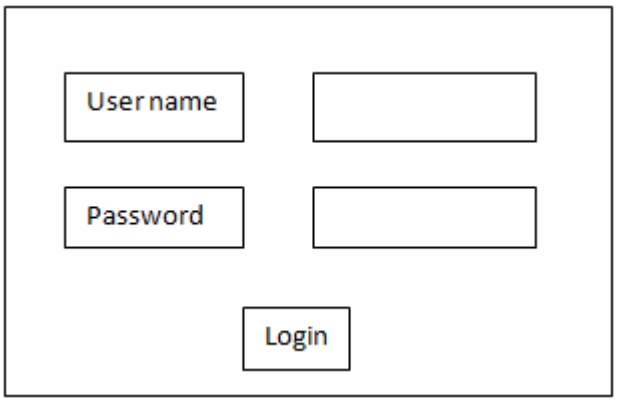
The page for relative's messages

9. Relative Write Message:

The page for relative to compose messages

The system, if looked at the big picture, is based on the transferring and monitoring of data. Thus presentation of data (consists of vital values of the patient) is of great importance, and website is the most suitable platform to succeed in that.

Below is a classical login page for the website.

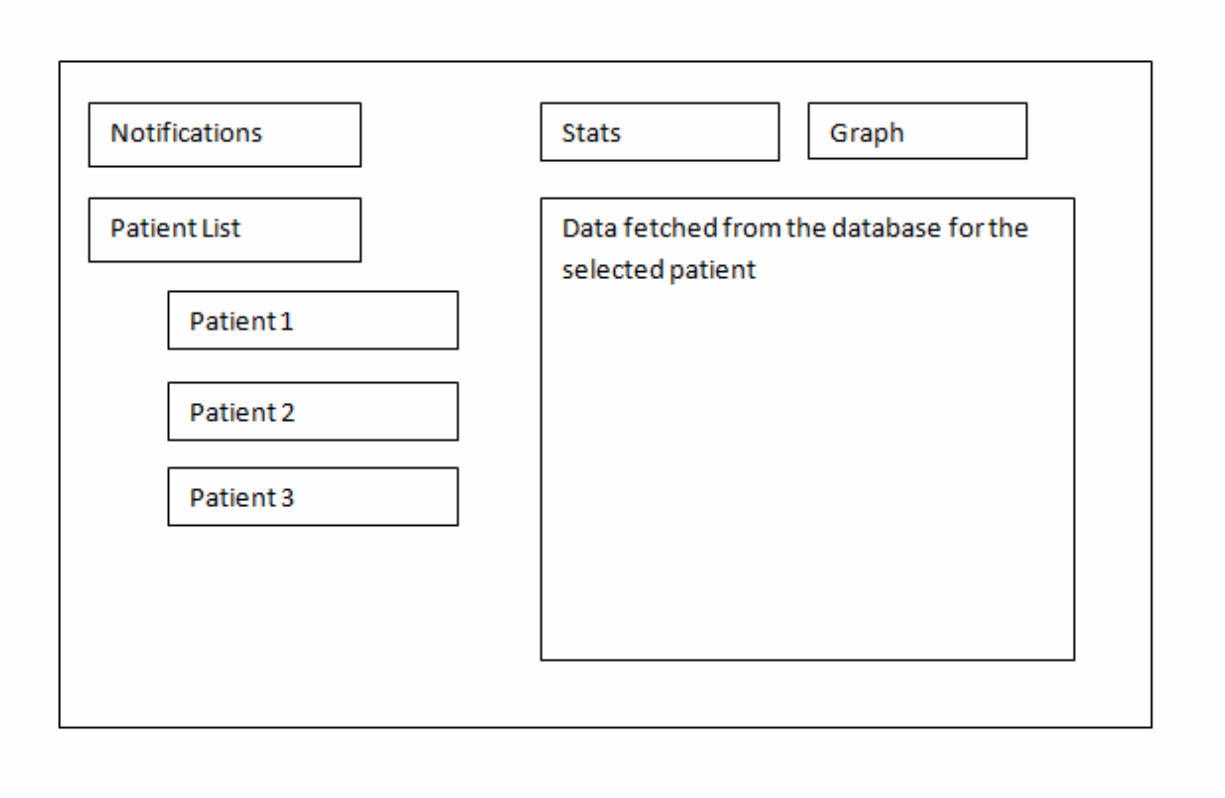


This page applies for both doctor and relative perspectives, but after login there will be differences between those two account types.

If the user is a patient or relative of a patient a screen similar to the one in the mobile program is displayed, since these users can monitor only one patient's data.

Otherwise if the account is owned by a doctor he/she will have more options. In the notifications section the doctor can view the patients with alarm conditions. It can also be displayed by selecting the patient but notifications will be quick link for the doctor.

Patient list will be expandable/shrinkable so that it will not take a large place in the page if desired. The right frame of the page will also be similar to the page which patient/relatives view. Data fetched from the database can be displayed either in numbers or graphs.



3. Storing Emergency Situation

When an emergency situation is handled, a copy of the emergency value will be added to the emergency table of our database. Handling this situation will be done by comparing the current value with the normal values. If current value is not in the normal range, it will be named as an emergency value.

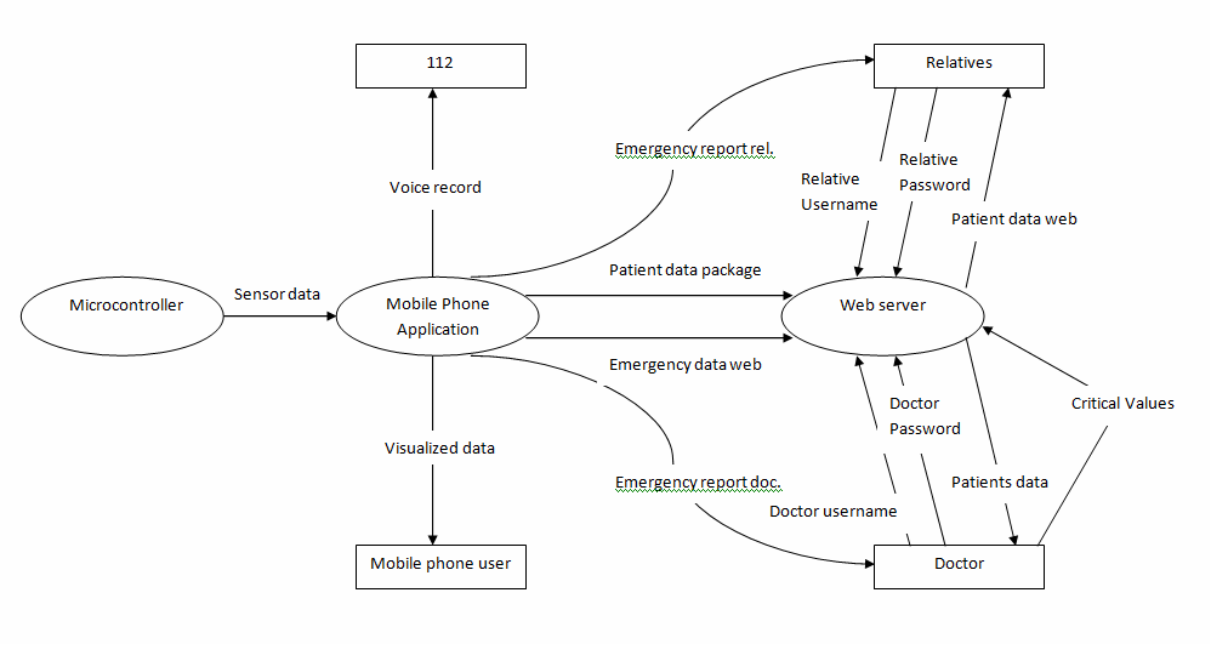
When the user clicks the “show emergency values” button in his page, the emergency table will be shown in the page. For the relative, this button will be in his/her homepage. For the doctor, this will be in the doctor’s ‘show patient’ page.

4. **General Features**

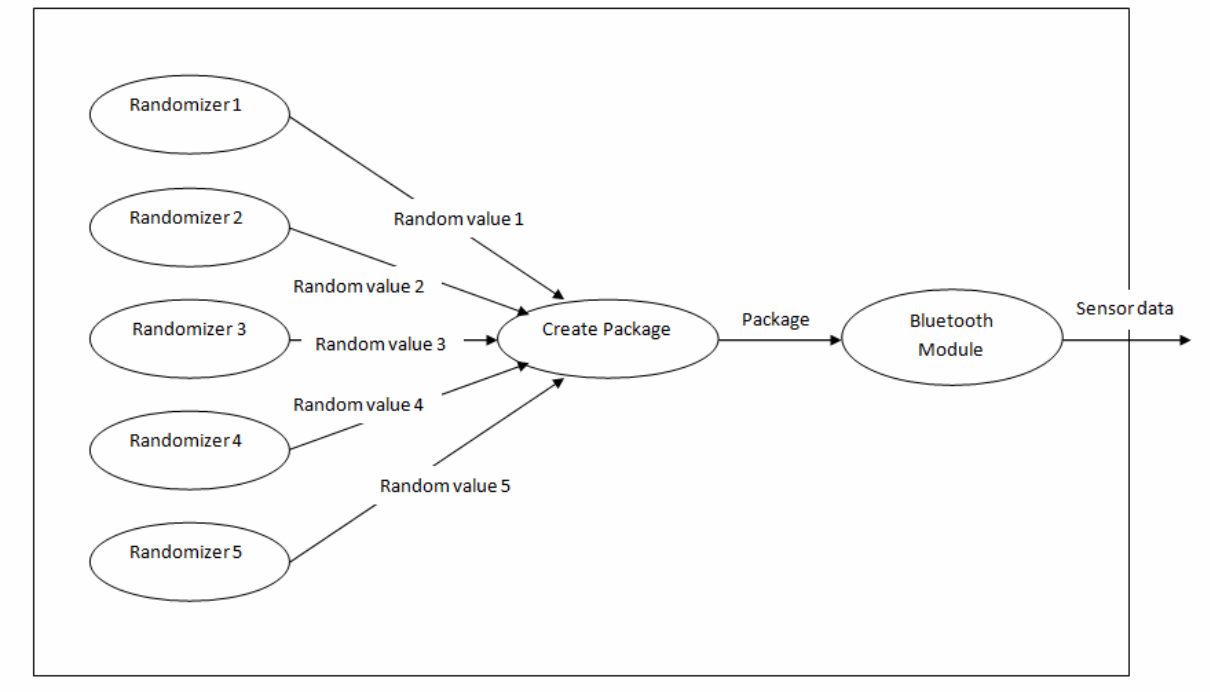
- Data from database will only go to the dynamic page when the user login. After that, when the user refresh the page, the new values will be shown. Else, the values won't go to the dynamic page.
- We are making the arrangements only with respect to the date, because the most important thing is the current situation and the current values of the patient. The aim of this web page is informing people about a patient's health, so we should provide the most important thing "the most recent values" by arranging with respect to date.
- The web page will support multi-chosen-values. The user can get only the values that he chose. It is necessary because some patients can have problems only about one type of value and the user may want to follow only that value.
- There will be an emergency history, because the user (no matter it is a relative or a doctor) may want to reach the last emergency situation, or the frequency of these situations. We can easily reach them from our emergency table and show them as an emergency history.
- There is a message part to make a connection between doctor and the relatives of patient. From this part, doctor or relative can send messages to each other to discuss the situation of the patient.

DIAGRAMS

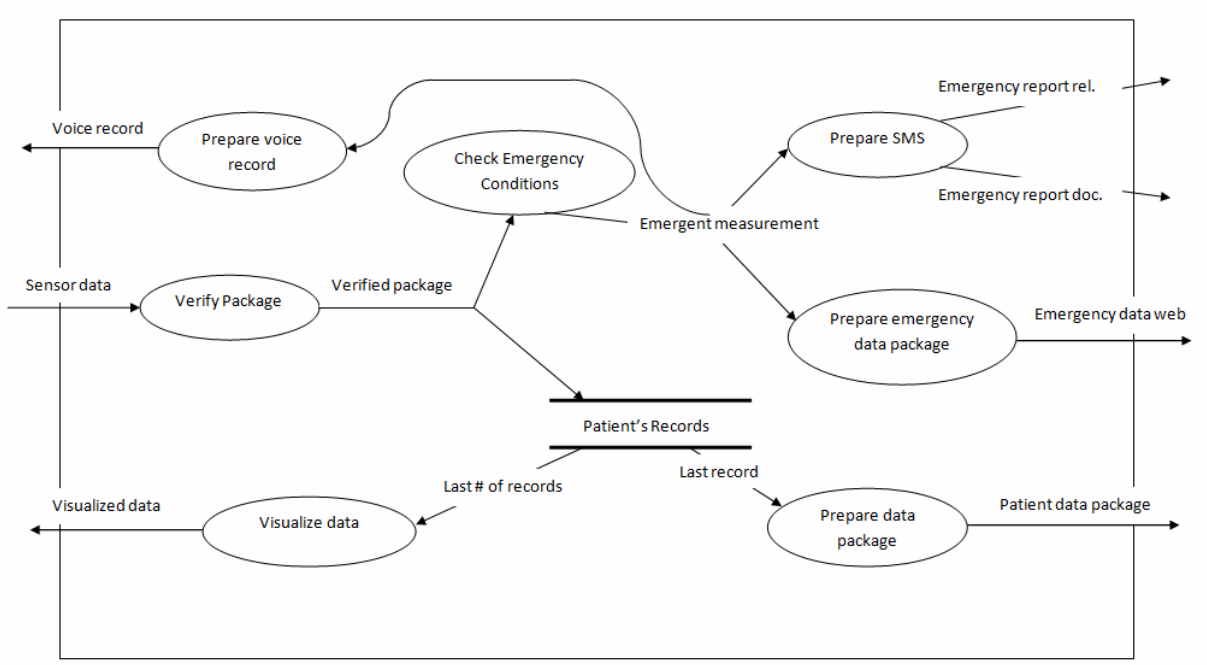
Data Flow Diagrams



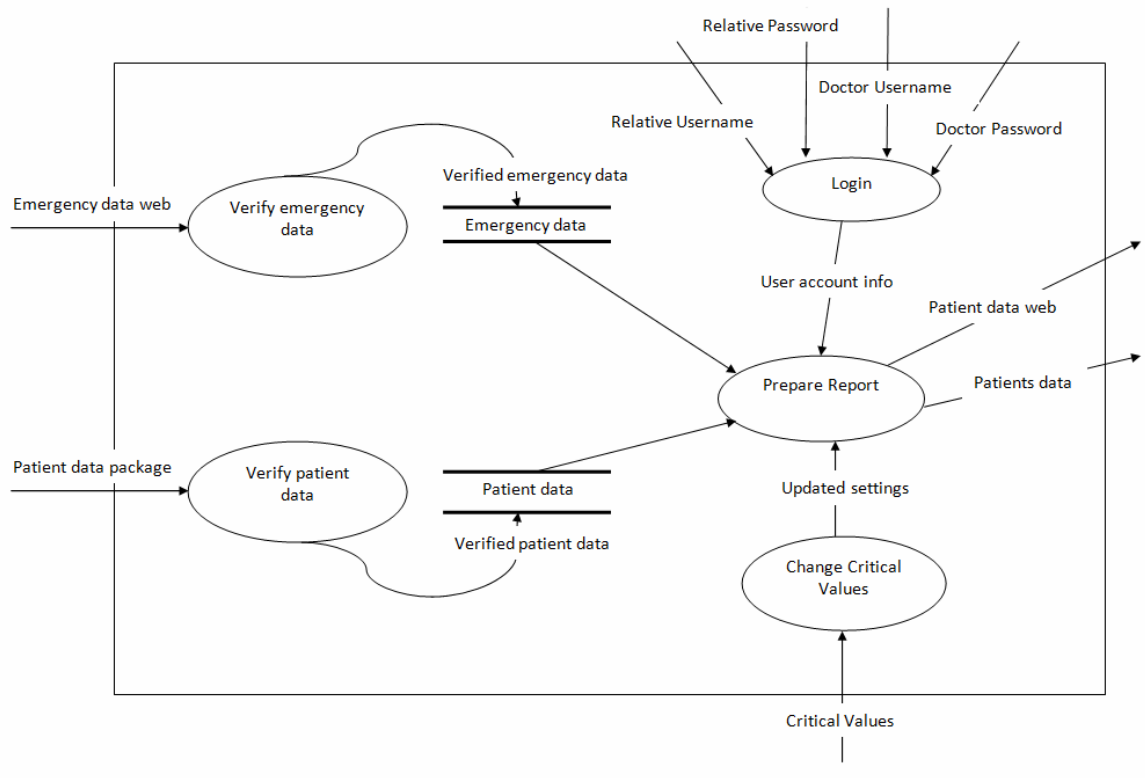
Level 0 DFD: Overall System



Level 1 DFD: Microcontroller

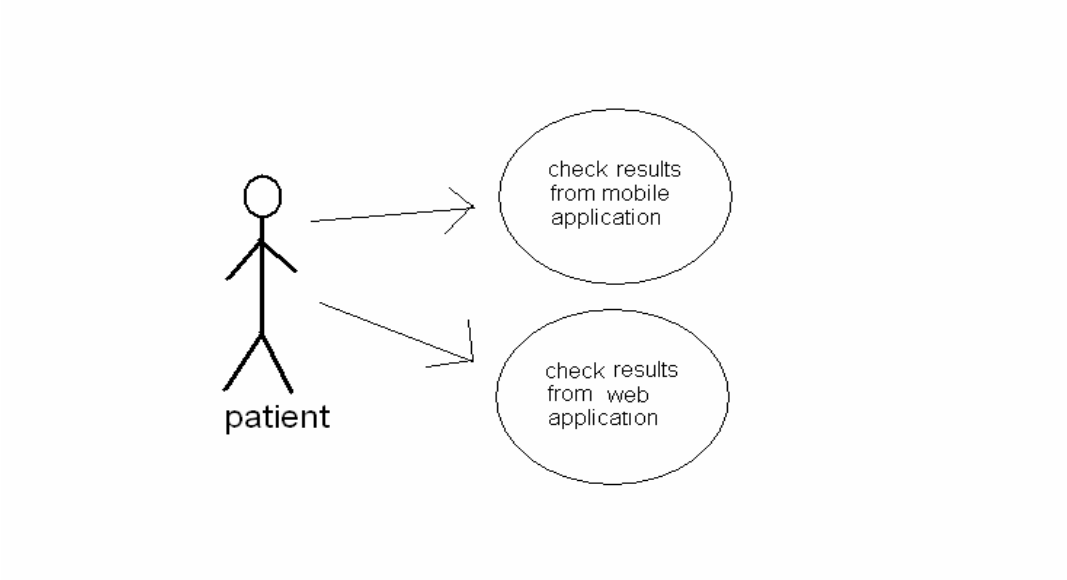


Level 1 DFD: Mobile Phone Application

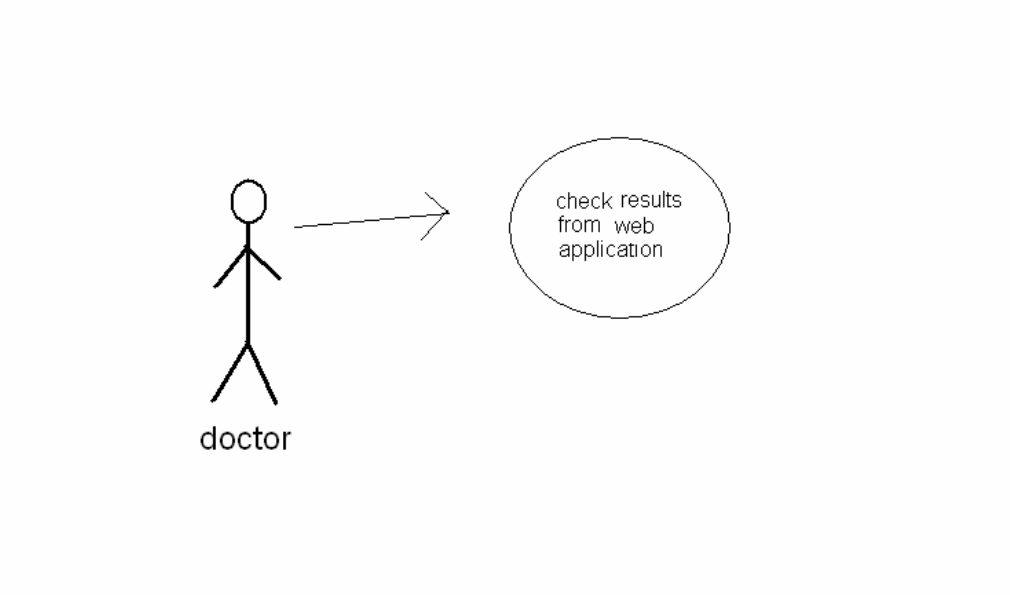


Level 1 DFD: Webserver

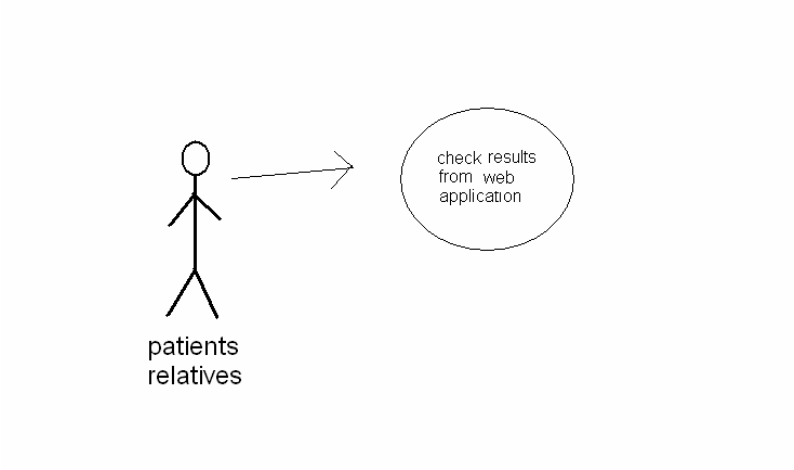
Use Case Diagrams



Patient Use Case Diagram

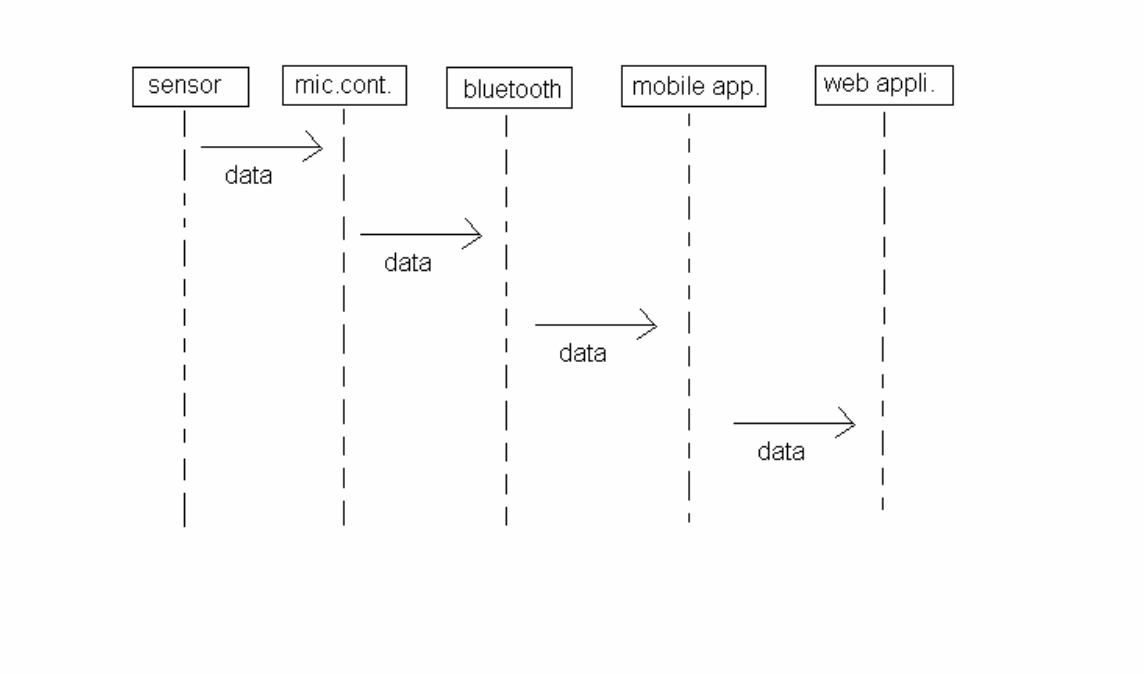


Doctor Use Case Diagram



Relatives Use Case Diagram

Sequence Diagram

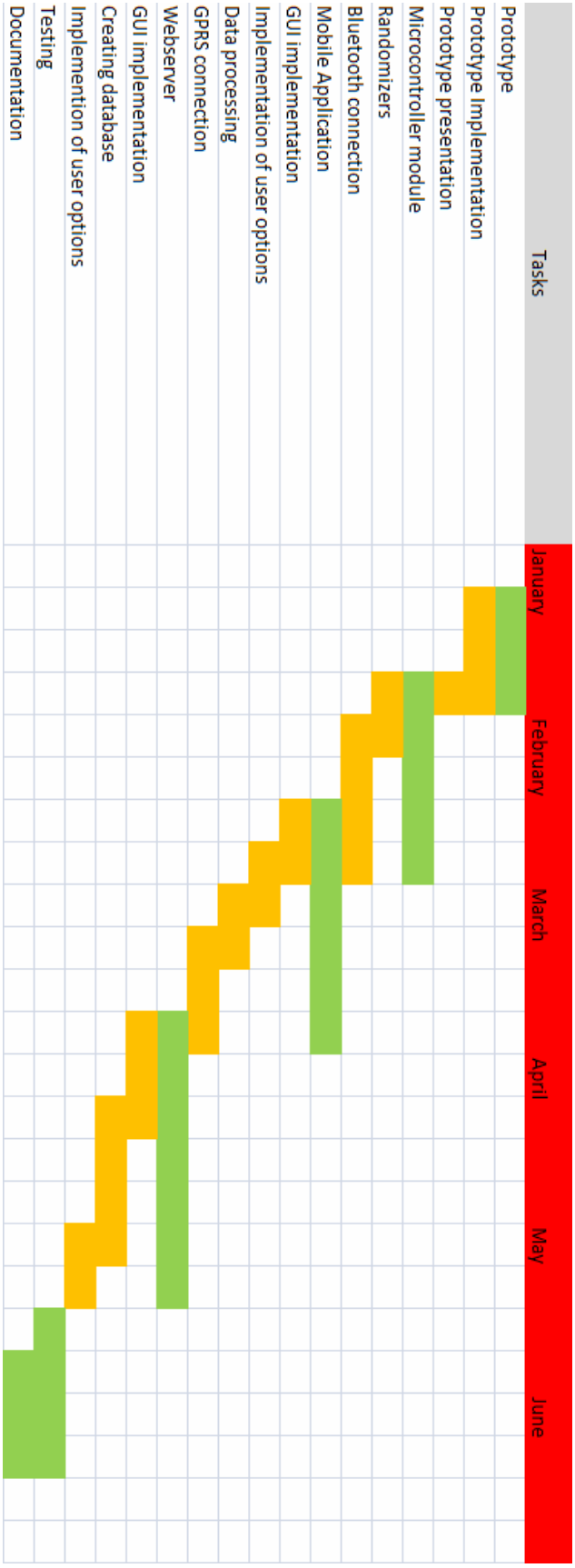


Sequence Diagram

PROJECT SCHEDULE

There are three phases mentioned in the design, which are the main parts of the system: microcontroller module, mobile application, web server. Unlike most other systems these parts can be implemented disregarding any order. However we are going to follow the same way that data flows through our system, while implementing the system.

Gantt Chart


Tasks
January
February
March
April
May
June
Prototype
Prototype Implementation
Prototype presentation
Microcontroller module
Randomizers
Bluetooth connection
Mobile Application
GUI implementation
Implementation of user options
Data processing
GPRS connection
Webserver
GUI implementation
Creating database
Implemention of user options
Testing
Documentation