

CHAPTER 1

INTRODUCTION

Telemedicine is the delivery of healthcare services through the use of Information and Communication Technologies (ICT) in a situation where the actors are not at the same location. The actors can be either health care professionals or a health care professional and a patient. It has the potential to improve the life of patients and empower them, support doctors in their work, improve the quality of care and patient safety. In the context of an ageing population, increased burden of chronic diseases, active participation of more demanding patients and ever increasing health expenditures, and the deployment of telemedicine services is important and urgent.

Diabetes mellitus, often simply referred to as diabetes—is a group of metabolic diseases in which a person has high blood sugar, either because the body does not produce enough insulin, or because cells do not respond to the insulin that is produced. This high blood sugar produces the classical symptoms of polyuria (frequent urination), polydipsia (increased thirst) and polyphagia (increased hunger).

In this project the biomedical signal, the blood sugar value is taken. The blood sugar value is acquired at the patient' side and the value is transmitted to Personal Computer (PC) and reconstructed. Then it is transmitted to the remote location using JAVA programming.

CHAPTER 2

LITERATURE REVIEW

2.1 GLUCOTRACK

GlucoTrack uses ultrasonic, conductivity and heat capacity technologies to non-invasively measure glucose levels in the blood. The device is battery-operated and includes a Main Unit (UMU), which contains display and control features, as well as transmitter, receiver and processor, and a **Personal** Ear Clip (PEC), which contains sensors and calibration electronics, and is clipped to the earlobe. The device is small, light and easy handling.

The patented simultaneous operation of three technologies is the unique concept behind our product. The measurements produced by the three technologies are fed into a proprietary algorithm, and the final result is displayed to the patient on a digital screen.

The patient can preset the device to issue visual and/or sound alerts if the measurements reach or surpass pre-determined values.

GlucoTrack is intended for use by Type 1 or Type 2 diabetics, gestational diabetes as well as patients with pre-diabetics, or Impaired Glucose Tolerance (IGT).

The Main Unit can be shared by up to three patients, although each will require a Personal Ear Clip. The device includes a USB port for downloading data for off-line analysis.

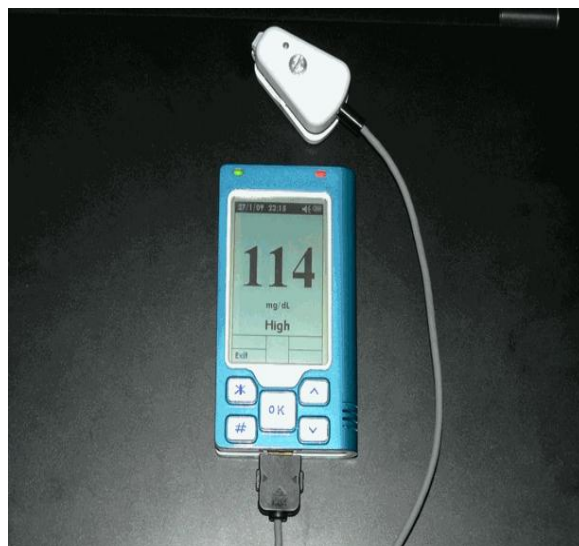


Fig 2.1 - GlucoTrack®, Model DF-F

2.2 GROVE INSTRUMENTS

Successful management of diabetes requires diabetic individuals to monitor their blood glucose levels at the frequency recommended by their physician. The American Diabetes Association reports that the typical person with diabetes tests at less than half the recommended frequency. Studies indicate that the pain, inconvenience, mess and expense of blood-based glucose measurement devices are the most important causes for sub-optimal monitoring. The consequences of insufficient monitoring are clear - patients may not achieve the proven benefits of good glycemic control and healthcare costs skyrocket. (Healthcare costs for diabetic individuals are five times the average).



Fig 2.2 – Portable glucose meter

2.3 SMSI® Glucose Sensor (in development)

The typical glucose-monitoring regimen for diabetes patients involves piercing a finger to obtain a blood sample, which is generally collected on a test strip and then analyzed in a suitable device. The current standard of care recommends that this procedure be repeated a minimum of four times per day. Because of the uncomfortable, cumbersome nature of this test regimen, compliance is poor.

SMSI® is developing a new approach to glucose monitoring that we believe will be an improvement over existing methods. SMSI's glucose sensor, will be inserted under the skin in a short outpatient procedure. The

sensor is designed to automatically measure interstitial glucose every few minutes, without any user intervention. The sensor implant will communicate wirelessly with a small external reader allowing the user to monitor glucose levels continuously or on demand. The reader is designed to be able to track the rate of change of glucose levels and warn the user of impending hypo- or hyperglycemia. The target operational life of the sensor implant will be 6-12 months, after which it would be replaced.

It is believe that diabetes patients will perceive this system to be an improvement in glucose monitoring. Clinical trials are designed to demonstrate that this system is accurate enough to aid people with diabetes in managing the disease. With positive results from our initial clinical studies, supplemental clinical studies are designed to demonstrate that the system is highly accurate, safe and easy to use. SMSI expects this will translate into significant adoption rates and improved patient health.

CHAPTER 3

SYSTEM STUDY

3.1 SOFTWARE REQUIREMENTS AND SPECIFICATION

3.1.1 PURPOSE OF THE PROJECT

The major purpose of the project is to aid the Doctor to monitor the Blood sugar value of their patients periodically and to make they comfort enough to compare the blood sugar value of particular patients for a particular period of time using a comparison chart.

3.1.2 SCOPE OF THE PROJECT

The project can be used in hospitals where patients can be monitored without any intrusion into their day-to-day activities by the doctor. Further our project can also be used by the relatives of the patient like son, daughter etc to keep an eye on the patients periodically.

3.1.3 OVERVIEW

Our system is a economical system that can measure the blood sugar value of the patient and transmit to the remote system using RF transmitter. RF receiver consumes the data and mails it to the doctor. At he receiving end the doctor fetches the mail using the provided application and monitors the patients condition accordingly. Once the mail is fetched it is automatically stored in the database and further manipulations are carried out.

3.2 GENERAL DESCRIPTION

Our project mainly focuses on monitoring the patient's sugar level using the glucometer circuit and transmitting it to the end user (doctor) using RF module. At the receiving end the data is received via mail and stored in the database by means of which periodically the doctor can take care of his patients and prescribe him accordingly.

3.2.1 EXISTING SYSTEM

The existing system in the market is just a glucometer by means of which we can find the blood glucose reading but cannot transmit or make the

doctor aware the reading periodically and moreover there are some systems that transmit data remotely to the doctor but they are not economical or efficient.

3.2.2 LIMITATION

1. It is not economical.
2. It is not compact.
3. It doesn't transmit the data automatically. It requires manual interaction
4. Limited storage of data about the patient.

3.2.3 PROPOSED SYTEM

We are about to propose a economical system that can measure the blood sugar value of the patient and transmit to the remote system using RF transmitter. RF receiver consumes the data and mails it to the doctor. At he receiving end the doctor fetches the mail using the provided application and monitors the patients condition accordoingly. Once the mail is fetched it is automatically stored in the database and further manipulations are carried out.

3.2.4 OBJECTIVE

Our goal is to develop a system to measure, record, and perform analysis on the glucose level of diabetics on a homebound basis. The device and corresponding software will be able to measure the blood sugar value of the diabetes patient when it is taken, and wirelessly transmit it to be saved as medical records. This is done by measuring the blood glucose using the device, followed by transmitting the measured data to the homebound computer using RF transceiver and then sending the data to an email address specified by the doctor, and accessing it through software running on the remote computer that is used to hold all the patients' information.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 INTRODUCTION

First Dr. Valdes from the UIC College of Medicine speak in class about his idea for a system to wirelessly monitor medical patients, we felt like it would be a great project since we could help out people in the world while gaining relevant experience in electrical engineering and project management. From there, we narrowed the focus of the project to wirelessly monitoring the glucose levels of diabetics since diabetes is a growing and costly problem worldwide. Essentially, the system will function similar for diabetics as On Star does for motor vehicles. It will log the patients' glucose level readings and send them to a medical database so that their doctor can monitor their levels and identify trends to enhance the treatment they are able to provide. All in all, just being able to help out many people all over the world is enough motivation for us to work on this exciting project.

Benefits to End Customer:-

- Patient glucose level will be logged automatically
- Doctor will automatically be sent an email with glucose levels within seconds of the level being tested, including a time and date stamp
- Patient glucose level history will be stored electronically
- Wireless design allows for an active lifestyle

Product Features

- Uses glucometer circuit to find the blood sugar value and encodes the data using HT12E
- Encoded data is transmits data using RF module
- Data is received and decoded using HT12D
- Decoded data is given to microcontroller where the data is regenerated
- The regenerated data is given to the system via serial port and mailed to the doctor

4.2 BLOCK DIAGRAM DESCRIPTION

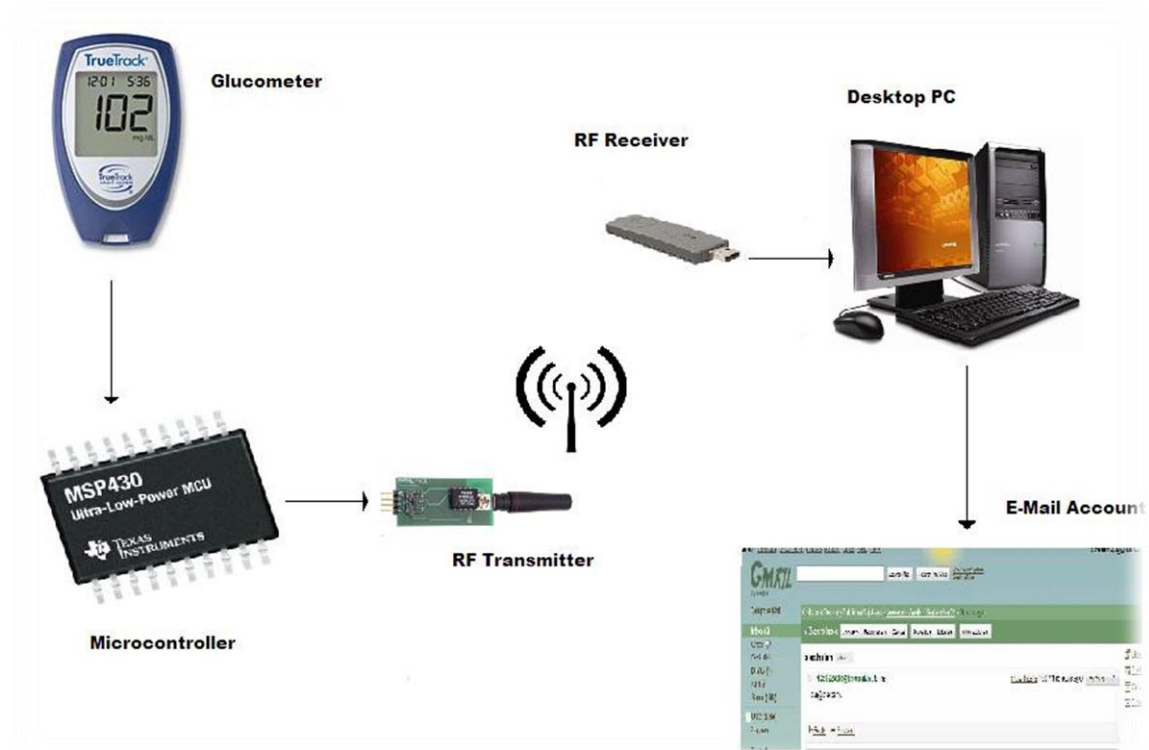


Figure 4.1 Block Diagram

The Glucometer circuit calculates the blood sugar value from the blood sample and gives the data in analog form to ADC which digitizes the data and gives it to the microcontroller. The data is then encoded and sent through RF transmitter.

The data transmitted is received at the other end using RF receiver. The received value is passed to a decoder which decodes the data and gives the output to a microcontroller. The microcontroller is programmed in such a way that it reconstructs the original data by performing some shift operations and finally transmits the data to the system via serial port.

Once the data is received in the system via com port it is transmitted to the doctor's email id with the patient's id so that the doctor can recognize the patient. The patient id is stored in the system by default. Once the doctor receives the reading periodically he prescribes his patients accordingly.

CHAPTER 5 TRANSMITTER SECTION

5.1 GLUCOMETER

The device that will collect glucose level readings (in mg/dL) through drops of blood from the diabetes patient when they normally test it, typically 3-7 times a day for a patient with type I diabetes. The glucometer in this case is a True Track Blood Glucose Monitoring System bought at a local drug store.

Test Strips

The blood sample is placed in the test strips and a reference voltage is given at the pair of electrodes available at the end of the strips. As a result of which current is conducted based on the blood sugar present in the blood. The current generated is in analog form. So it is passed to ADC0808 for further processing.

5.2 ANALOG TO DIGITAL CONVERTER ADC0808

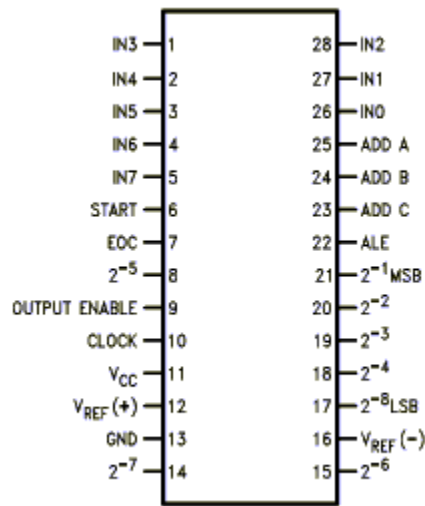


Fig 5.1 ADC0808

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended

analog signals. The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE[®] outputs. The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For 16- channel multiplexer with common output (sample/hold port) see ADC0816 data sheet. (See AN-247 for more information.)

Features

- Easy interface to all microprocessors
- Operates ratio metrically or with 5 VDC or analog span adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package
- ADC0808 equivalent to MM74C949
- ADC0809 equivalent to MM74C949-1

Key Specifications

- Resolution 8 Bits
- Total Unadjusted Error $\pm 1/2$ LSB and ± 1 LSB
- Single Supply 5 VDC
- Low Power 15 mW
- Conversion Time 100 ms

Absolute Maximum Ratings

Supply Voltage (VCC) (Note 3) 6.5V

Voltage at Any Pin ± 0.3 V to (VCC ± 0.3 V)

Except Control Inputs

Voltage at Control Inputs ± 0.3 V to ± 15 V

(START, OE, CLOCK, ALE, ADD A, ADD B, ADD C)

Storage Temperature Range b65 C to a150 C
Package Dissipation at TAe25 C 875 mW
Lead Temp. (Soldering, 10 seconds)
Dual-In-Line Package (plastic) 260 C
Dual-In-Line Package (ceramic) 300 C
Molded Chip Carrier Package
Vapor Phase (60 seconds) 215 C
Infrared (15 seconds) 220 C
ESD Susceptibility 400V

Operating Conditions

Temperature Range (Note 1) TMINsTAsTMAX
ADC0808CJ b55 CsTAa125 C
ADC0808CCJ, ADC0808CCN,
ADC0809CCN b40 CsTAa85 C
ADC0808CCV, ADC0809CCV b40 C s TA s a85 C
Range of VCC (Note 1) 4.5 VDC to 6.0 VDC

5.3 POWER SUPPLY INTRODUCTION

Most electronic circuits require DC voltage sources or power supplies. If the electronic device is to be portable, then one or more batteries are usually needed to provide the DC voltage required by the electronic circuits.

But batteries have limited life span and cannot be recharged. The solution is to convert the alternating current household line voltage to DC voltage source.

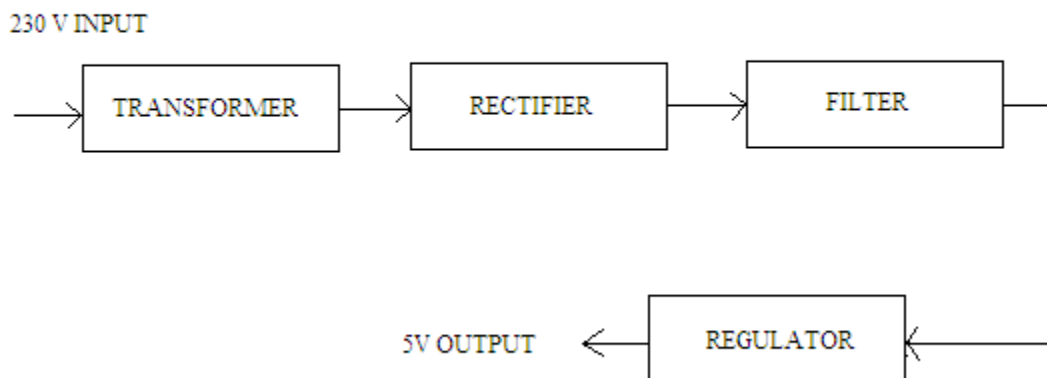


Fig 5.2 Block diagram of power supply

The block diagram of AC to DC power supply consists of

1. Transformer : Steps the household line voltage up or down as required
2. Rectifier : Converts AC voltage to pulsating DC voltage
3. Filter : Smooth the pulsating DC voltage to a varying DC voltage
4. Regulator: Fix the output voltage to a constant value.

5.3.1 Regulators

The simplest regulator is a large capacitor in parallel with the load. The capacitor stores DC voltage while the load voltage increases to its peak value. The capacitor converts the pulsating DC voltage of the rectifier into a smooth DC load voltage.

Two important parameters of a capacitor regulator are its working voltage and its capacitance. The working voltage must be at least equal to no-load output voltage of the power supply. The capacitance determines the amount of ripple that appears on the DC output when current is drawn from the circuit. The amount of ripple decreases with the increase in capacitance.

7805 and 7905 Regulators

The 78xx series of devices is a family of self contained fixed linear voltage regulator integrated circuits. The 78xx family is a very popular choice for many electronic circuits which require a regulated power supply, due to their ease of use and relative cheapness. When specifying individual ICs within this family, the xx is replaced with a two-digit number, which indicates the output voltage the particular device is designed to provide(for example, the 7805 has a 5 volt output, while the 7812 produces 12 volts). The 78xx line is positive voltage regulators, meaning that they are designed to produce a voltage that is positive relative to a common ground. There is a related line of 79xx devices which are complementary negative voltage regulators. 78xx and 79xx ICs can be used in combination to provide both positive and negative supply voltages in the same circuit, if necessary.

5.4 AT89S52 MICROCONTROLLER

5.4.1 Architecture of AT89S52

8051 consist of central processing unit, two kinds of memory, IO ports, the mode status and data registers, random logic needed for a variety

of peripheral functions. These elements communicate through an 8bit data bus which runs throughout the chip referred as internal data bus.

Central Processing Unit

CPU of 8051 consist of 8bit arithmetic and logic unit with associated registers like A,B,PSW,SP, the 16bit PC and the data pointer registers.

Arithmetic and Logic Unit

The arithmetic and logic unit performs 8bit arithmetic and logical operations over the operand held by the temporary registers. Users cannot access these registers.

Accumulator It is an 8bit register which holds a source operand and receives the result of arithmetic instructions. The accumulator can be the source or destination for logical operations and number of special data movement instructions, including look-up tables and external RAM expansion.

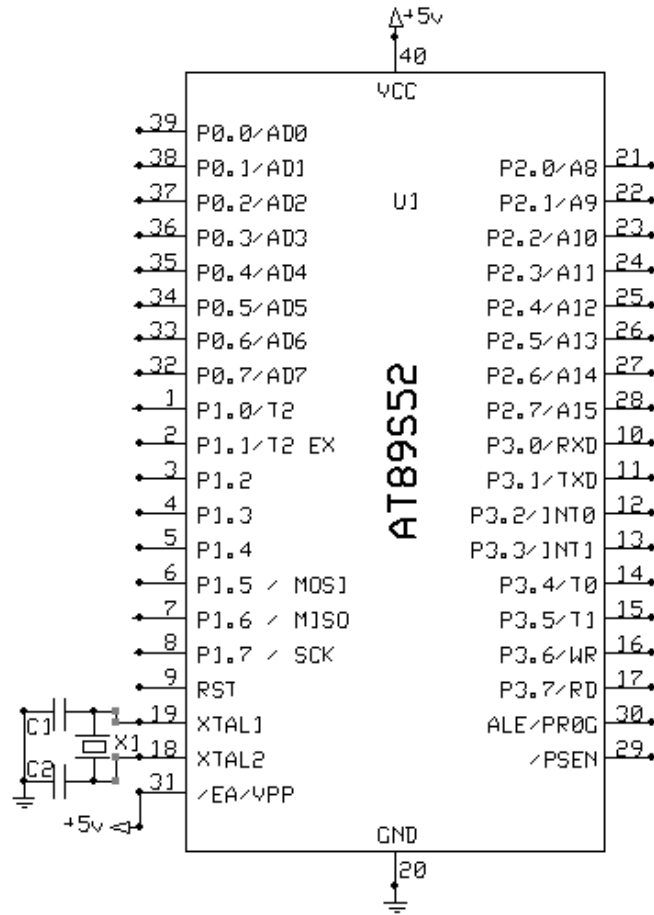


Fig 5.3 Pin diagram of AT89S52

5.4.2 PIN DESCRIPTION

5.4.2.1 VCC

Supply voltage

5.4.2.2 GND

Ground

5.4.2.3 PORT 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs. Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pull-ups are required during program verification.

5.4.2.4 PORT 1

Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table. Port 1 also receives the low-order address bytes during Flash programming and verification

5.4.2.5 PORT 2

Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that uses 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that uses 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

5.4.2.6 PORT 3

Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL) because of the pull-ups. Port 3 receives some control signals for Flash programming and verification. Port 3 also serves the functions of various special features of the AT89S52

5.4.2.7 RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives high for 98 oscillator

periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

5.4.2.8 ALE/PROG

Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory. If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

5.4.2.9 PSEN

Program Store Enable (PSEN) is the read strobe to external program memory. When the AT89S52 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory

5.4.2.10 EA/VPP

External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming

5.4.2.11 XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

5.4.2.12 XTAL2

Output from the inverting oscillator amplifier.

5.5 RF TRANSMITTER

5.5.1 Overview

The STT-433 is ideal for remote control applications where low Cost and longer range is required. The transmitter operates from a 1.5-12V supply, making it ideal for battery-powered applications. The transmitter employs a SAW-stabilized oscillator, ensuring accurate frequency control for best range performance. Output power and harmonic emissions are easy to control, making FCC and ETSI compliance easy. The manufacturing-friendly SIP style package and low-cost make the STT-433 suitable for high volume applications.

5.5.2 Features

- 433.92 MHz Frequency
- Low Cost
- 1.5-12V operation
- 11mA current consumption at 3V
- Small size
- 4 dBm output power at 3V

5.5.3 Applications

- Remote Keyless Entry (RKE)
- Remote Lighting Controls
- On-Site Paging
- Asset Tracking
- Wireless Alarm and Security Systems
- Long Range RFID
- Automated Resource Management

5.5.4 Specification

Parameter	Symbol	Min	Typ.	Max	Unit
Operating Voltage	Vcc	1.5	3.0	12	Volts DC
Operating Current Data = VCC	Icc	-	11mA @3V 59mA @5V	-	mA
Operating Current Data = GND	Icc	-	100	-	uA
Frequency Accuracy	TOL fc	-75	0	+75	Khz
Center Frequency	Fc	-	433	-	Mhz
RF Output Power		-	4 dBm@3V (2 mW) 16 dBm@5V (39 mW)		dBm / mW
Data Rate		200	1K	3K	BPS
Temperature		-20		+60	Deg. C
Power up delay			20		ms

Table 5.1 Specification

5.5.5 Pin Description

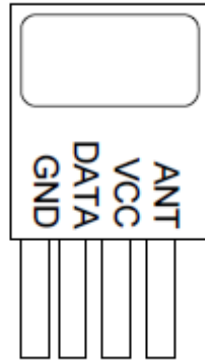


Fig 5.4 RF Transmitter

Pin Name	Description
ANT	50 ohm antenna output. The antenna port impedance affects output power and harmonic emissions. An L-C low-pass filter may be needed to sufficiently filter harmonic emissions. Antenna can be single core wire of approximately 17cm length or PCB trace antenna.
VCC	Operating voltage for the transmitter. VCC should be bypassed with a .01uF ceramic capacitor and filtered with a 4.7uF tantalum capacitor. Noise on the power supply will degrade transmitter noise performance.
DATA	Digital data input. This input is CMOS compatible and should be driven with CMOS level inputs.
GND	Transmitter ground. Connect to ground plane.

Table 5.2 Pin Description

CHAPTER 6

RECEIVER SECTION

6.1 RF Receiver

6.1.1 Overview

The STR-433 is ideal for short-range remote control applications where cost is a primary concern. The receiver module requires no external RF components except for the antenna. It generates virtually no emissions, making FCC and ETSI approvals easy. The super-regenerative design exhibits exceptional sensitivity at a very low cost. The manufacturing-friendly SIP style package and low-cost make the STR-433 suitable for high volume applications.

6.1.2 Features

- Low Cost
- 5V operation
- 3.5mA current drain
- No External Parts are required
- Receiver Frequency: 433.92 MHZ
- Typical sensitivity: -105dBm
- IF Frequency: 1MH

6.1.3 Applications

- Car security system
- Sensor reporting
- Automation system
- Remote Keyless Entry (RKE)
- Remote Lighting Controls
- On-Site Paging
- Asset Tracking
- Wireless Alarm and Security Systems
- Long Range RFID
- Automated Resource Management

6.1.4 Specification

Parameter	Symbol	Min	Typ.	Max	Unit
Operating Voltage	Vcc	4.5	5.0	5.5	VDC
Operating Current	Icc	-	3.5	4.5	mA
Reception Bandwidth	BW rx	-	1.0	-	MHz
Center Frequency	Fc	-	433.92	-	MHz
Sensitivity	-	-	-105	-	dBm
Max Data Rate	-	300	1k	3K	Kbit/s
Turn On Time	-	-	25	-	ms
Operating Temperature	T op	-10	-	+60	°C

Table 6.1 Specification

6.1.5 Pin Outs

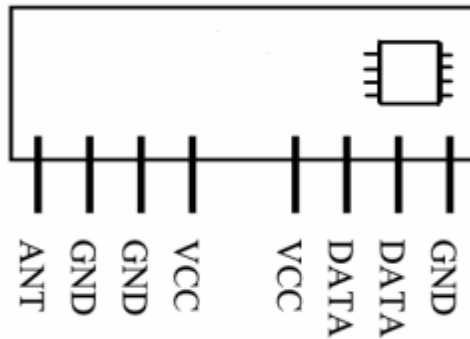


Fig 6.1 RF Receiver

Pin Name	Description
ANT	Antenna input.
GND	Receiver Ground. Connect to ground plane.
VCC(5V)	VCC pins are electrically connected and provide operating voltage for the receiver. VCC can be applied to either or both. VCC should be bypassed with a .1μF ceramic capacitor. Noise on the power supply will degrade receiver sensitivity.
DATA	Digital data output. This output is capable of driving one TTL or CMOS load. It is a CMOS compatible output.

Table 6.2 Pin Description

6.2 Serial Communication

The Serial Port is harder to interface than the Parallel Port. In most cases, any device you connect to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. On the software side of things, there are many more registers that you have to attend to than on a Standard Parallel Port. (SPP)

Devices which use serial cables for their communication are split into two categories. These are DCE (Data Communications Equipment) and DTE (Data Terminal Equipment.) Data Communications Equipment are devices such as your modem, TA adapter, plotter etc while Data Terminal Equipment is your Computer or Terminal. The electrical specifications of the serial port are contained in the EIA (Electronics Industry Association) RS232C standard. It states many parameters such as -

- A "Space" (logic 0) will be between +3 and +25 Volts.
- A "Mark" (Logic 1) will be between -3 and -25 Volts.
- The region between +3 and -3 volts is undefined.
- An open circuit voltage should never exceed 25 volts. (In Reference to GND)
- A short circuit current should not exceed 500mA. The driver should be able to handle this without damage. (Take note of this one!)

Above is nowhere near a complete list of the EIA standard. Line Capacitance, Maximum Baud Rates etc are also included. For more information please consult the EIA RS232-E standard. It is interesting to note however, that the RS232C standard specifies a maximum baud rate of 20,000 BPS!, which is rather slow by today's standards. Revised standards, EIA-232D & EIA-232E were released, in 1987 & 1991 respectively. Serial Ports come in two "sizes", There are the D-Type 25 pin connector and the D-Type 9 pin connector both of which are male on the back of the PC, thus you will require a female connector on your device. Below is a table of pin connections for the 9 pin and 25 pin D-Type connectors.

CHAPTER 7 TESTING

7.1 DATABASE MANAGEMENT SYSTEM

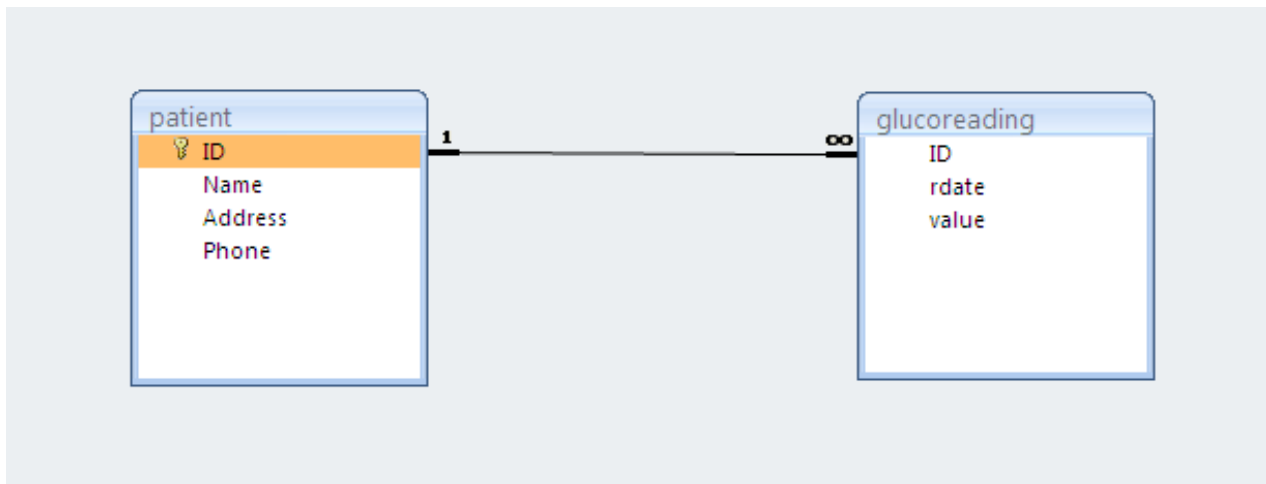
The database we have used is MS Access to store patient information and their blood sugar readings. The database contains three tables called patient, glucoreading and admin.

Patient (id (text), name (text), address (text), phone (text));

Glucoreading (id (text), rdate (date/time), value (number));

Admin (user (text), pass (text));

There is a direct relationship between patient and glucoreading table as follows



7.2 TESTING

Software testing involves with two grouping namely

- Black box testing
- White box testing

7.2.1 BLACK BOX TESTING

This approach focuses on inputs, outputs and principle function of a software module. In this project ,the software inputs are tested by giving different data types and by entering different ranges of values and

check whether if it violates constraints or not. This testing is done through module by module. Black box testing is also called as Functional testing. It involves random testing, domain testing and cause effect grapping in dynamic approach whereas in static approach it is done as specification provided.

7.2.2 WHITE BOX TESTING:

This approach looks into the structure of code for a software module; white box testing is also called as glass box testing. This testing involves with computation testing, domain testing, path based testing, data generation and mutation analysis in dynamic approach. In case of static approach, code walkthroughs, inspections, program proving, symbolic execution and anomaly analysis,

When a number is entered in the place of a character a message box asking us to enter correct value.

If some values are inserted in a table then it shows us a message box to enter all values.

If a character is entered in the place of the number then it asks us to enter correct value.

CHAPTER 8

EXPERIMENTAL RESULTS

8.1 STEPS INVOLVED IN THE PROJECT

In the Patient's side, plug-in the receiver to the PC's serial port. Place the test strip between the two electrodes of the transmitter and place a drop of the blood sample whose sugar value is to be deduced. Start the client program in the PC to enable it read the transmitted value. Then the program automatically mails the reading to the doctor.

Now at the doctor's PC, start the doctor's program and click the Get Mail button to retrieve the mail. The program automatically stores the value in the database. Now select a patient from the list and click Get Info to retrieve the information about the patient. Select the duration over which the sugar readings are to be analyzed and click Get Info button to get a bar chart of the readings recorded.

8.2 RESULTS

8.2.1 CIRCUITS



Fig 8.1 Blood Sample



Fig 8.2 Test Strip



Fig 8.3 Transmitter

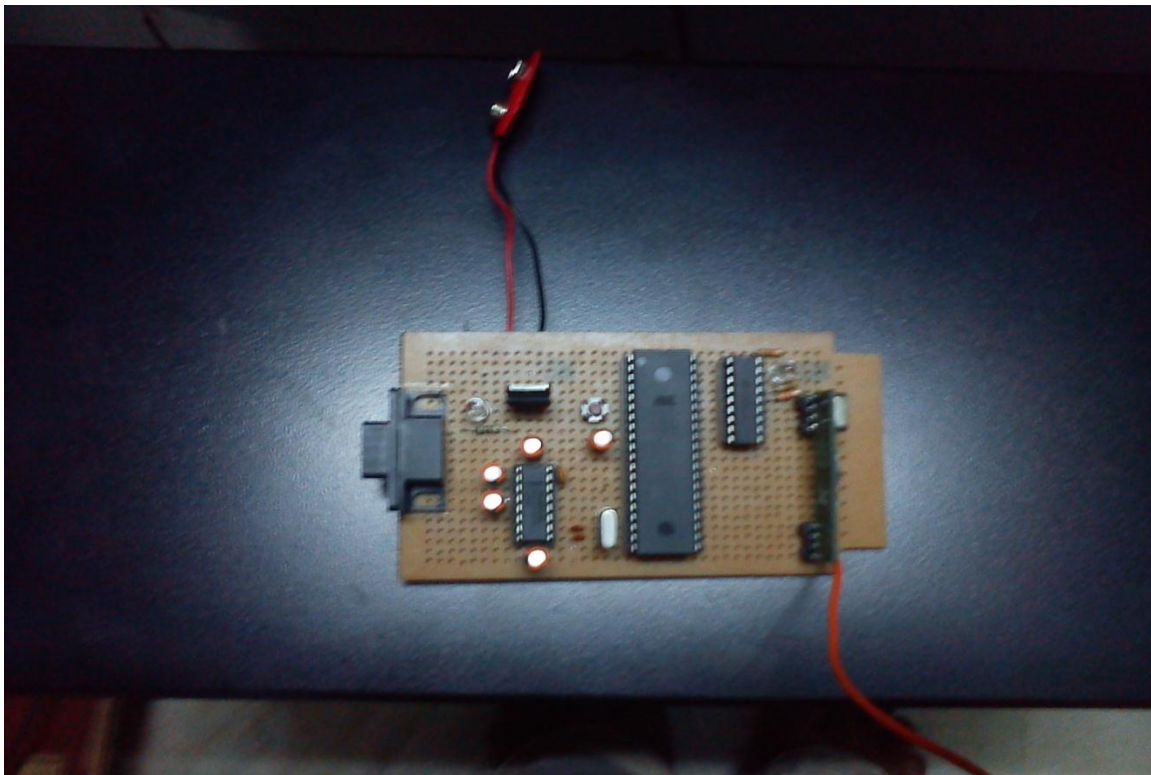


Fig 8.4 Receiver

8.2.2 JAVA APPLICATION

Blood Sugar Monitoring Using Pervasive Computing Tec...

Register

Patient Name :

Medical ID :

E-Mail ID :

Fig 8.5 Patient Registration form

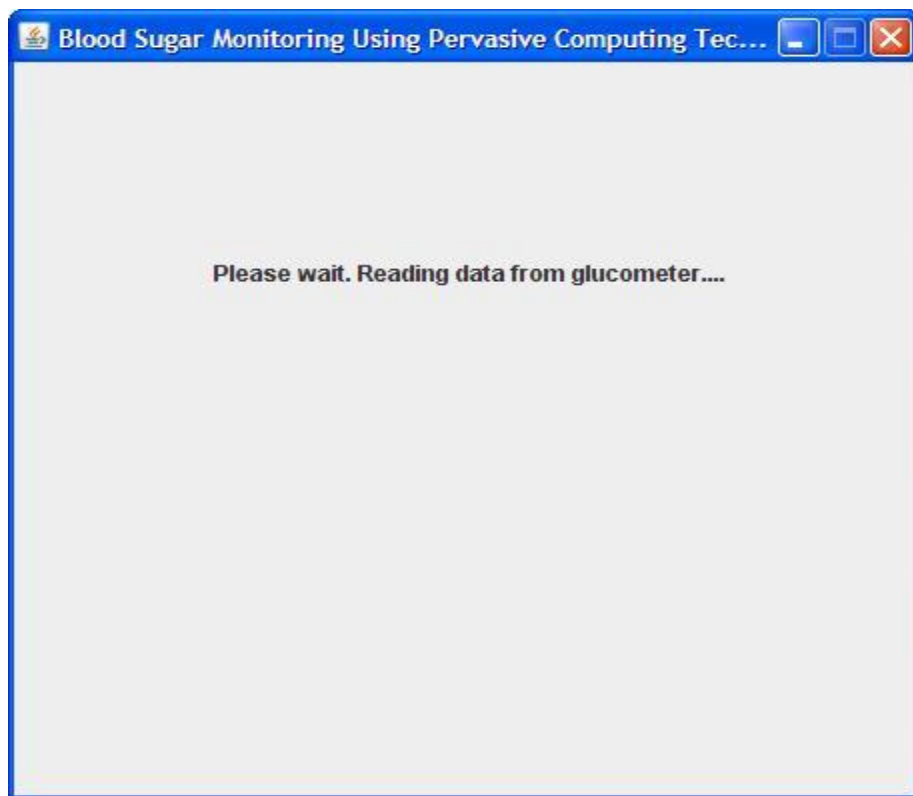


Fig 8.6 Waiting for data

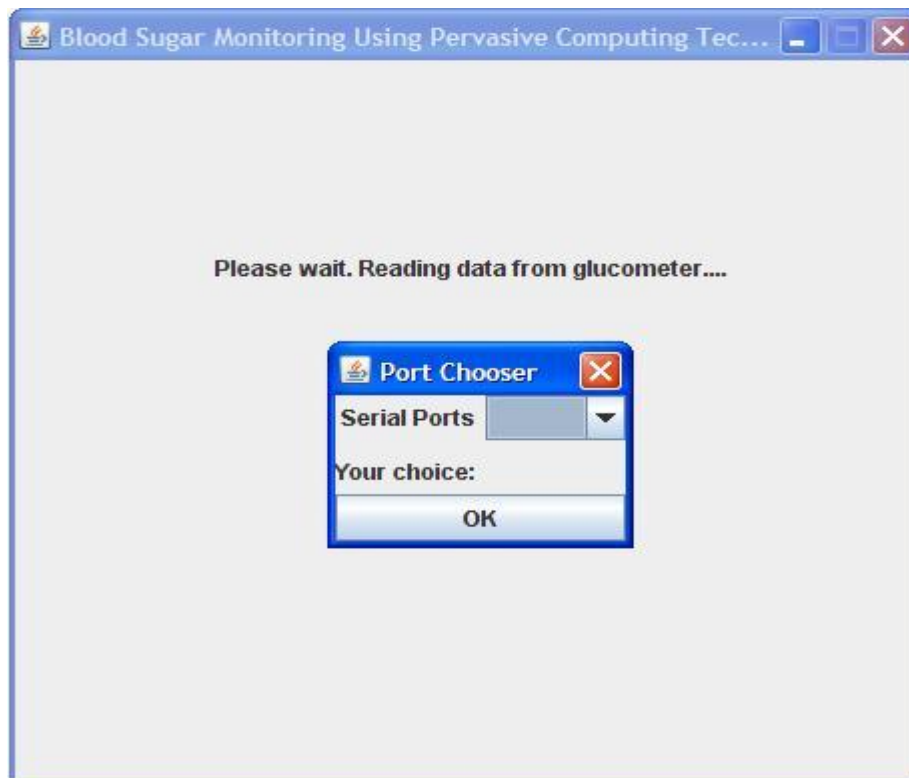


Fig 8.7 Port Selection

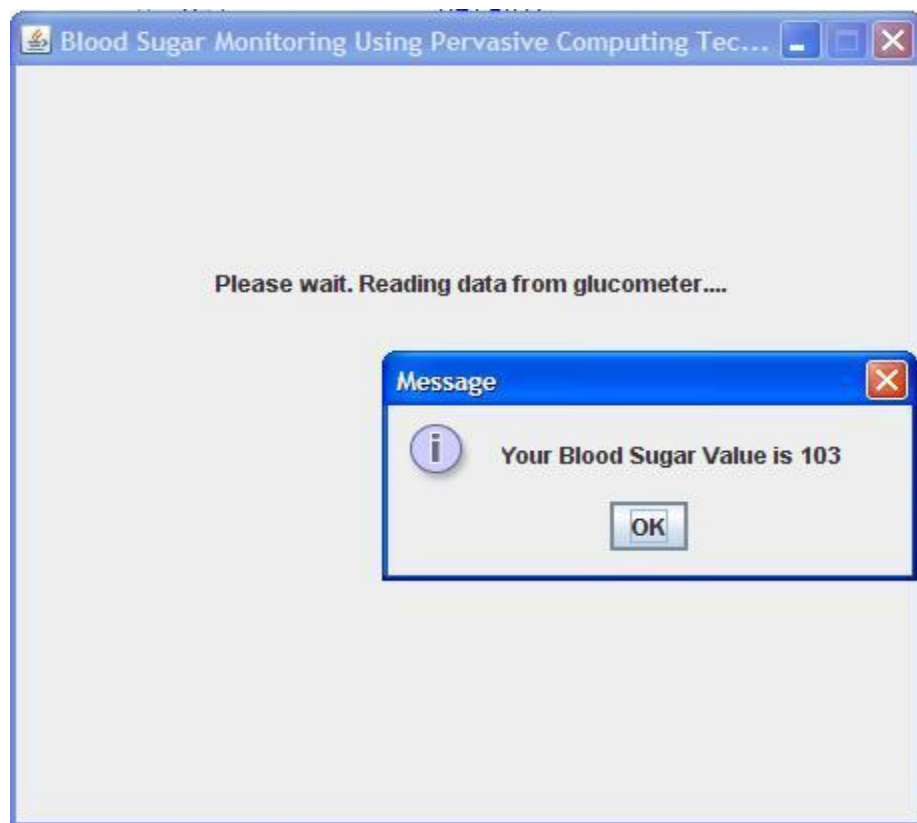


Fig 8.8 Sugar Value Display

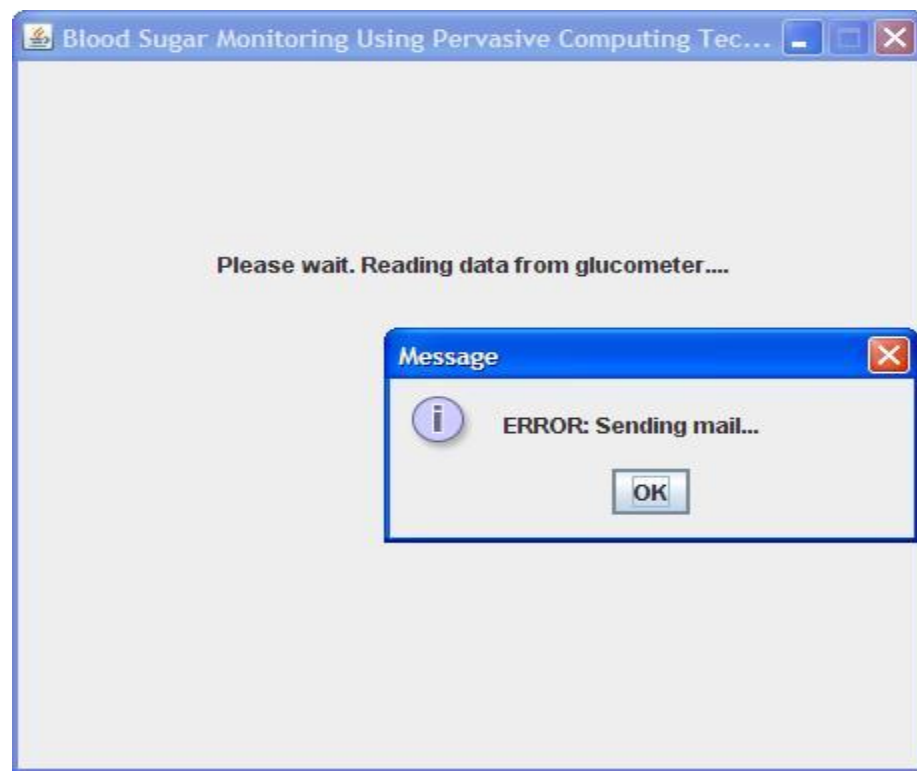
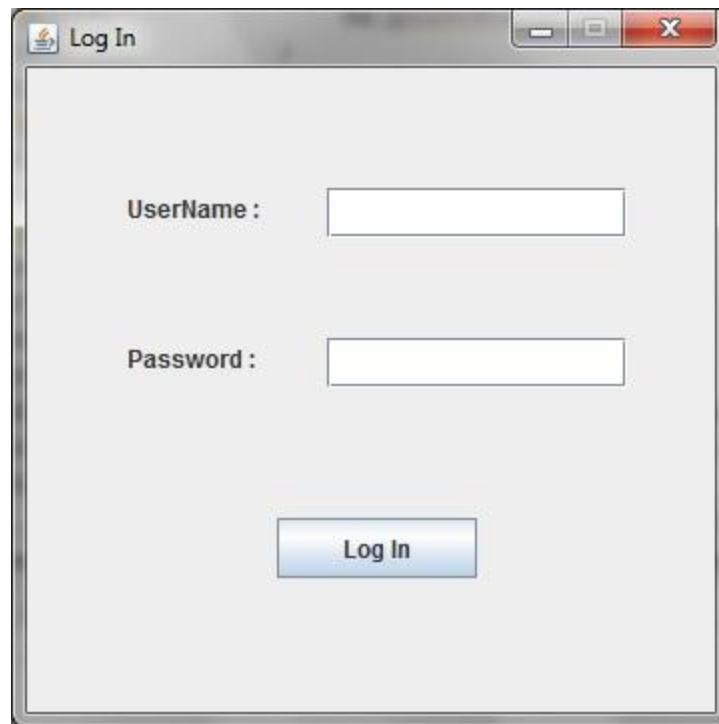
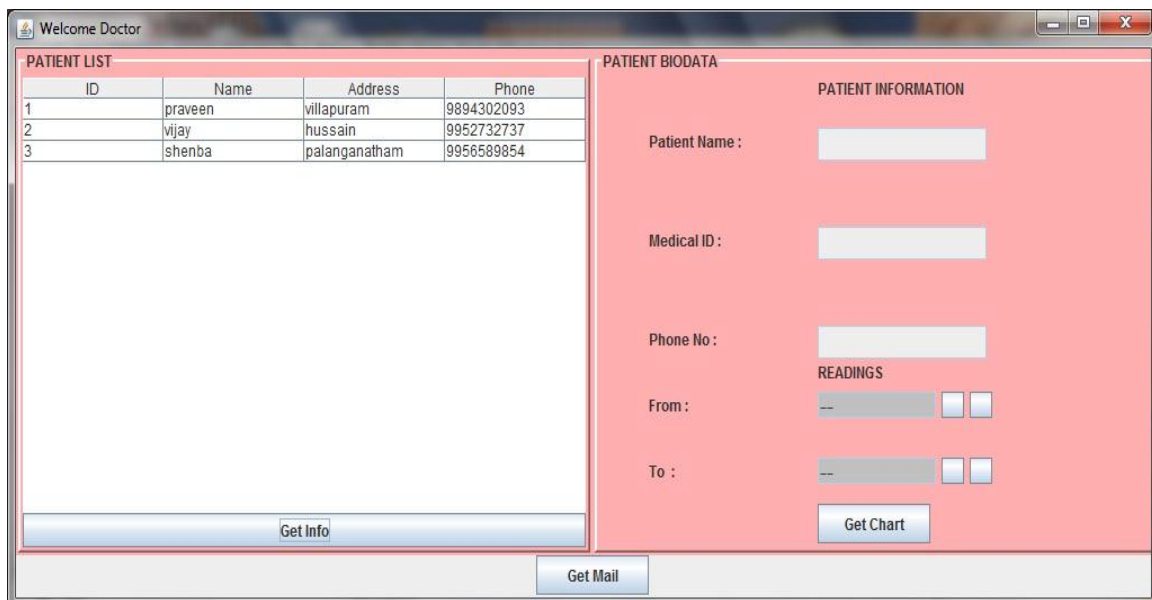


Fig 8.9 Error Report



A screenshot of a 'Log In' window. It features a title bar with a small icon and the text 'Log In'. The window contains two input fields: 'UserName :' and 'Password :'. Below these fields is a 'Log In' button.

Fig 8.10 Log In Screen



A screenshot of a 'Welcome Doctor' window. It is divided into two main sections: 'PATIENT LIST' on the left and 'PATIENT BIODATA' on the right. The 'PATIENT LIST' section contains a table with patient information. The 'PATIENT BIODATA' section contains input fields for patient information and a 'Get Chart' button. At the bottom of the window is a 'Get Mail' button.

ID	Name	Address	Phone
1	praveen	villapuram	9894302093
2	vijay	hussain	9952732737
3	shenba	palanganatham	9956589854

PATIENT BIODATA

PATIENT INFORMATION

Patient Name :

Medical ID :

Phone No :

READINGS

From :

To :

Fig 8.11 Patient Description

Welcome Doctor

PATIENT LIST

ID	Name	Address	Phone
1	praveen	villapuram	9894302093
2	vijay	hussain	9952732737
3	shenba	palanganatham	9956589854

Get Info

PATIENT BIODATA

PATIENT INFORMATION

Patient Name : praveen

Medical ID : 1

Phone No : 9894302093

READINGS

From : 01-Mar-2011

To : 29-Apr-2011

Get Chart

Get Mail

Fig 8.12 Bar Chart Request

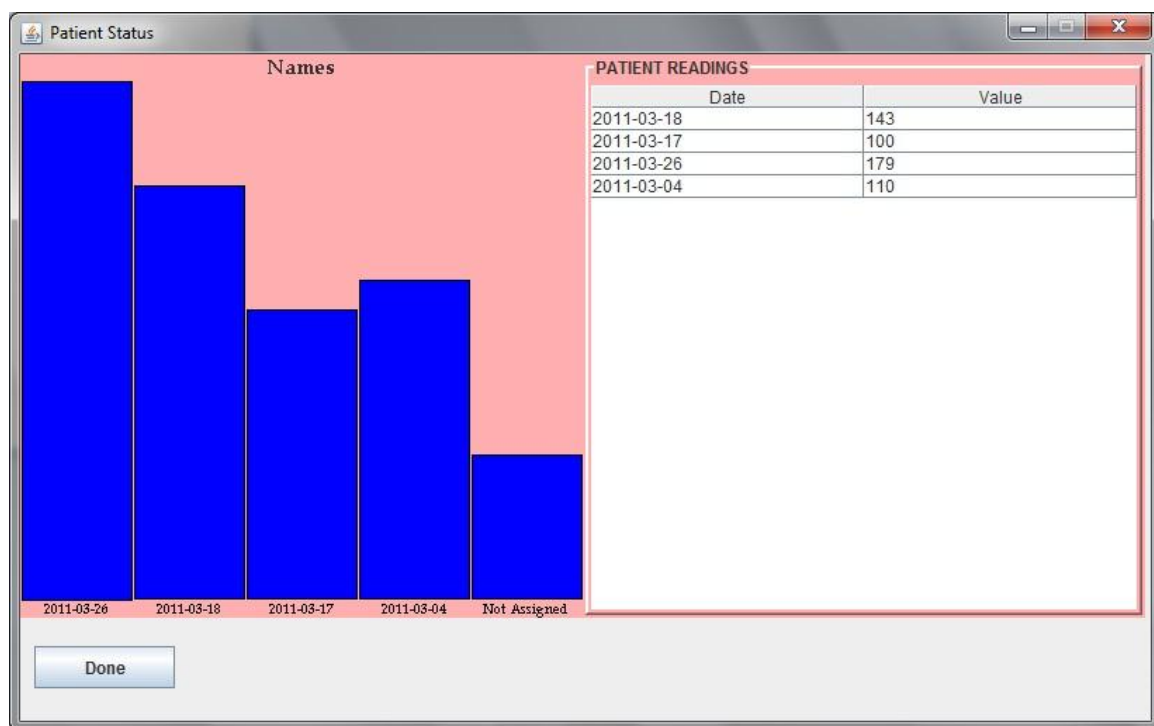


Fig 8.13 Bar Chart Analysis

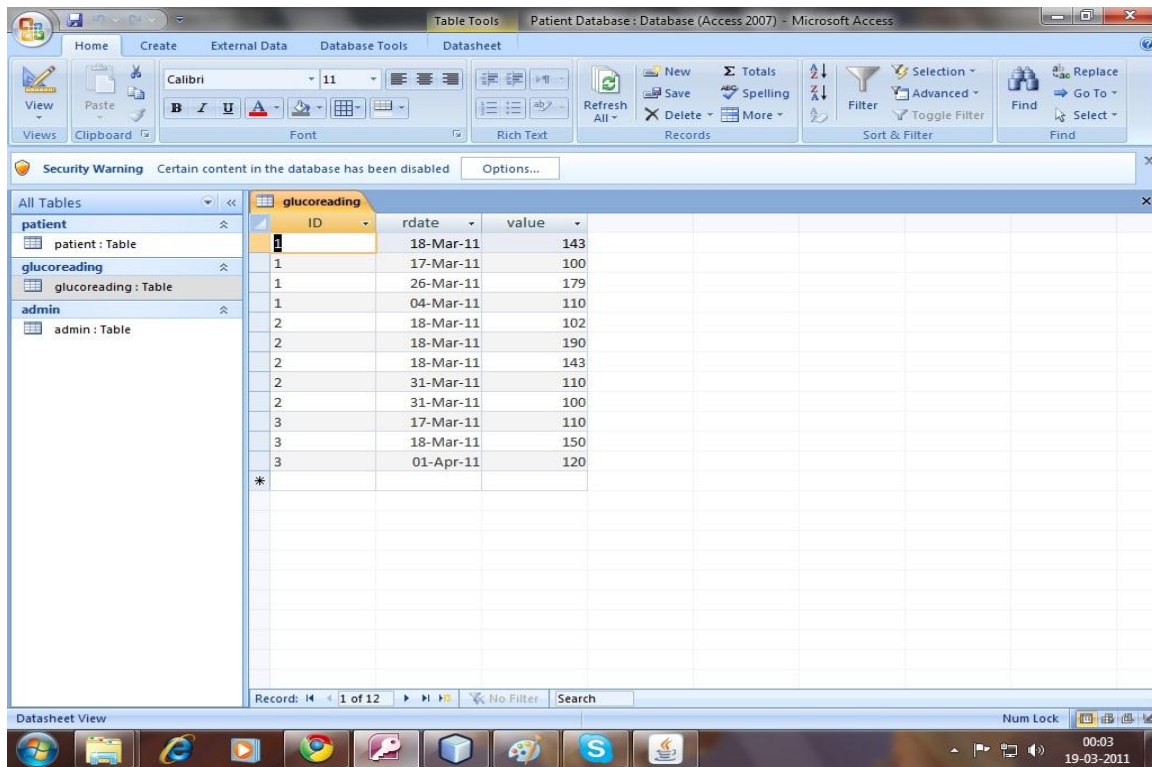


Fig 8.14 Initial Database

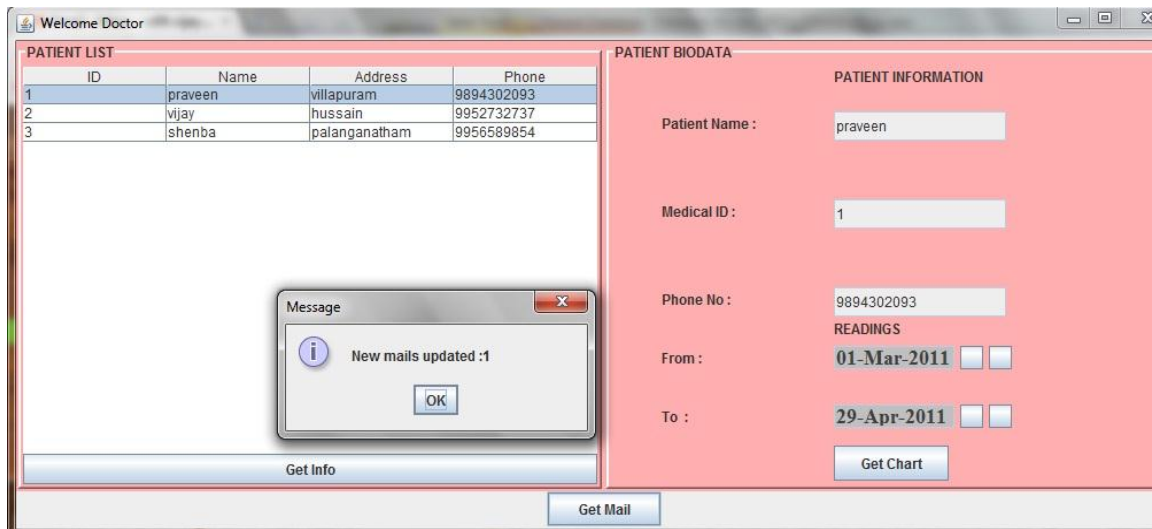


Fig 8.15 Fetching Mails

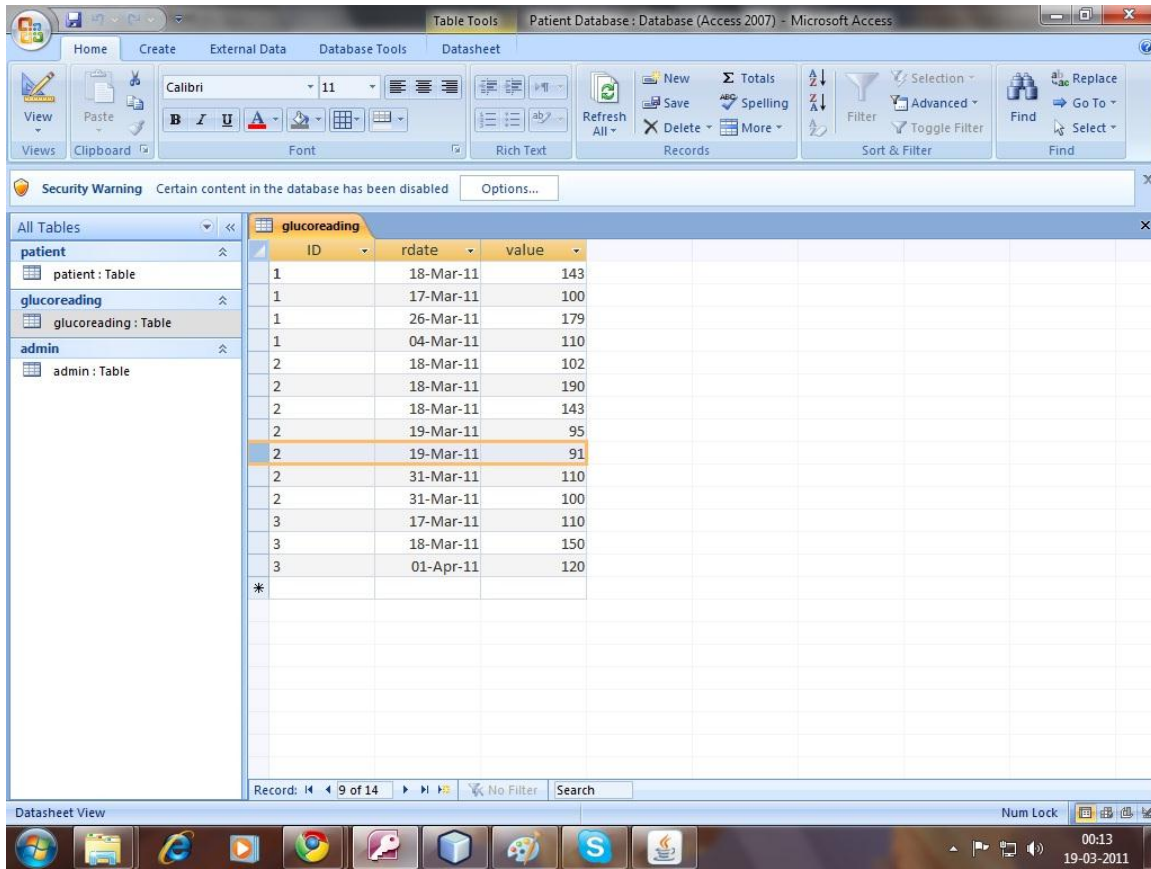


Fig 8.16 Updated Database

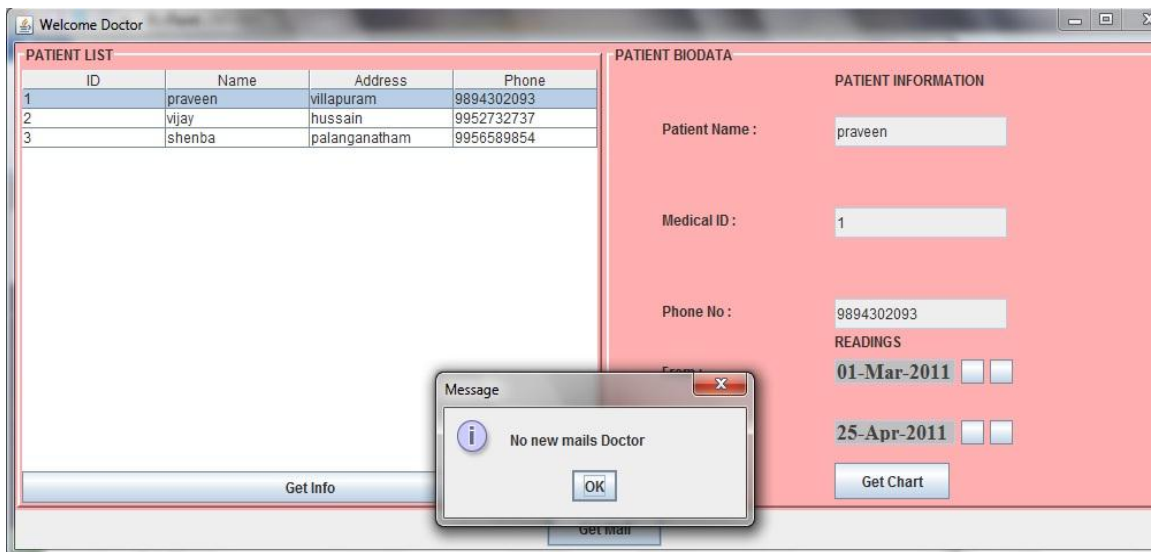


Fig 8.17 Mail Report

CHAPTER 9

SOCIAL IMPACT

1. This system helps to remotely monitor patient's blood sugar.
2. It can be used by non professional people and rural people to measure the blood sugar with ease
3. Cost Effective
4. Ubiquitous Health Monitoring at home.
5. Has greater impact in the fields of Telemedicine and Tele-Homecare.

CHAPTER 10

CONCLUSION

Home care represents a growing field in the health assistance. It reduces cost and increases the quality of life of patients. As the modern life becomes more stressful and acute diseases appear, prolonged treatments become more necessary. Home care offers the possibility of treatment in patient's house, with the assistance of the family.

Thus we have designed a REMOTE BLOOD SUGAR MONITORING SYSTEM wherein efficient utilization of energy and time is established. We did it using Glucometer circuit and then using microcontroller to transmit the appropriate reading to the serial port of the computer using RF transmitter. At the receiver end RF receiver is used to get the data, decode it and give it to the microcontroller to convert into 8bit before giving it to the serial port. After received at the serial port, the data is detected and mailed to the doctor wherein it is manipulated accordingly.

The readings are acceptably accurate. The operations of the device are reliable and have not produced any major problems.

Recent studies conclude that early and specialized pre-hospital management contributes to emergency case survival. The system can also be applied in emergency telemedicine applications in reason of its mobility.

FUTURE ENHANCEMENT

This project can be further improved by making the project compact. Some components can be substituted to make it much smaller and efficient. This would help the product easily portable. Also many other new applications based on the database can be designed.

CHAPTER 11

BIBLIOGRAPHY

1. <http://www.diabetes.org/>
2. Robert, J., "Continuous Monitoring of Blood Glucose," *Hormone Research*, vol. 57, pp. 81-4, 2002
3. <http://www.diabetesmonitor.com/meters.html>
4. The topic of Multicore processors could be the way to spur medical innovation while supporting legacy applications. *The journal of new electronics* 27 Oct 2009.
5. The topic of Wireless connectivity is set to streamline processes and reduce costs in the medical world. *The journal of new electronics* 27 Oct 2009.
6. The white paper on Wireless Glucose Monitoring System by Brian Alongi, Alex Hsieh on sept 25, 2009.

CHAPTER 12

APPENDIX

MAILING

The mailing section consists of two parts namely.

- Sending Mail
- Mail Retrieval

Sending Mail

The client GUI program consists of option to send mail to the doctor once the reading is read through the serial port. The coding is as follows.

Mailer.java

```
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;
import java.io.*;

public class Mailer {

    String d_email = "vs04.vijay@gmail.com",
        d_password = "*****",
        d_host = "smtp.gmail.com",
        d_port = "465",
        m_to = "pervasivetechno@gmail.com",
        m_subject = "";
    m_text="";

    public Mailer(String nw) {
        m_text = nw;
        try {
            FileInputStream fin = new FileInputStream("patinfo");
            int id = (fin.read());
            m_subject = "" + id;
            System.out.println(m_subject);
        } catch (Exception w) {
            System.out.println(w);
        }
    }

    public Mailer() {
```

```

Properties props = new Properties();
props.put("mail.smtp.user", d_email);
props.put("mail.smtp.host", d_host);
props.put("mail.smtp.port", d_port);
props.put("mail.transport.protocol", "smtps");
props.put("mail.smtp.starttls.enable", "true");
props.put("mail.smtps.auth", "true");
props.put("mail.smtps.debug", "true");
props.put("mail.smtps.socketFactory.port", d_port);
props.put("mail.smtps.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
props.put("mail.smtps.socketFactory.fallback", "false");

try {
    Authenticator auth = new SMTPAuthenticator();
    Session session = Session.getInstance(props, auth);
    session.setDebug(true);

    MimeMessage msg = new MimeMessage(session);
    msg.setText(m_text);
    msg.setSubject(m_subject);
    msg.setFrom(new InternetAddress(d_email));
    msg.addRecipient(Message.RecipientType.TO, new
InternetAddress(m_to));

    Transport tr = session.getTransport("smtps");
    tr.connect(d_host, 465, d_email, d_password);
    msg.saveChanges();
    tr.sendMessage(msg, msg.getAllRecipients());
    tr.close();
    javax.swing.JOptionPane.showMessageDialog(null, "Mail Sent");
    System.out.println("Mail Sent");
} catch (Exception mex) {
    System.out.println(mex);
    javax.swing.JOptionPane.showMessageDialog(null, "ERROR:
Sending mail...");
}
}

public static void main(String[] args) {

```

```

        Mailer blah = new Mailer("" + 5);
    }

    private class SMTPAuthenticator extends javax.mail.Authenticator {

        @Override
        public PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(d_email, d_password);
        }
    }
}

Pass.java
import javax.swing.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.lang.String;

public class pass extends JApplet
{
    JFrame pa;
    JTextField t1;
    JPasswordField t2,t3;
    JLabel l1,l2,l3;
    JButton confirm;
    Connection con;
    Statement st;
    ResultSet rs;
    String u,p,cp;
    public pass()
    {
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

            con=DriverManager.getConnection("jdbc:odbc:patient","","");
            st=con.createStatement();

            pa=new JFrame("CREATE ACCOUNT");

```

```
pa.setLayout(null);
pa.setVisible(true);
pa.setBounds(200,200,390,390);
pa.setResizable(false);
pa.setDefaultCloseOperation(pa.EXIT_ON_CLOSE);
```

```
l1=new JLabel("User Name ");
l1.setVisible(true);
l1.setBounds(50,50,100,40);
pa.add(l1);
```

```
l2=new JLabel("Password ");
l2.setVisible(true);
l2.setBounds(50,125,100,40);
pa.add(l2);
```

```
l3=new JLabel("Confirm Password ");
l3.setVisible(true);
l3.setBounds(50,200,150,40);
pa.add(l3);
```

```
t1=new JTextField();
t1.setVisible(true);
t1.setBounds(200,60,150,25);
pa.add(t1);
```

```
t2=new JPasswordField();
t2.setVisible(true);
t2.setBounds(200,135,150,25);
pa.add(t2);
```

```
t3=new JPasswordField();
t3.setVisible(true);
t3.setBounds(200,210,150,25);
pa.add(t3);
```

```
confirm=new JButton("Confirm");
confirm.setVisible(true);
```

```

confirm.setBounds(150,280,100,30);
pa.add(confirm);

confirm.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try{
            u=t1.getText();
            p=t2.getText();
            cp=t3.getText();
            if((!u.equals("")) && (!p.equals("")) &&
(!cp.equals("")))
            {
                if(p.equals(cp))
                {
                    st.executeUpdate("insert into
admin values('"+u+"','"+p+"')");

                    JOptionPane.showMessageDialog(null,"Authentications and
Tokens Validated");

                    pa.dispose();
                }
                else
                {

                    JOptionPane.showMessageDialog(null,"Please Confirm
password correctly");

                    t1.setText("");
                    t2.setText("");
                    t3.setText("");
                }
            }
            else
            {

                JOptionPane.showMessageDialog(null,"Please enter all the fields
");

                t1.setText("");

```



```

                t2.setText("");
                t3.setText("");
            }
        }
        catch(Exception e2)
        {

JOptionPane.showMessageDialog(null,e2);
        }
    }
});

    pa.setBounds(200,200,400,400);

}
catch(Exception e)
{
    JOptionPane.showMessageDialog(null,e);
}
}
public static void main(String ar[])
{
    pass p=new pass();
}
}

```

Proj.java

```

import javax.swing.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.border.*;
import java.awt.Toolkit;
import java.util.Timer;
import java.util.TimerTask;

public class proj extends JApplet
{
    public JFrame jf;

```

```

Timer timer;
int i,j;
int screenWidth;
JTextField pname,pid,pmid;
JLabel lname,lpid,lpmid,rd;
int screenHeight;
JPanel patinfo;
int w1,h1,x1,y1;
JButton submit;
public proj()
{
    try{
        GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice[] gs = ge.getScreenDevices();
        DisplayMode dm = gs[0].getDisplayMode();
        screenWidth = dm.getWidth();
        screenHeight = dm.getHeight();
        i=screenWidth;
        j=screenHeight;

        x1=screenWidth/4;
        y1=screenHeight/4;

        jf=new JFrame("Blood Sugar Monitoring Using Pervasive
Computing Technologies");
        jf.setLayout(null);
        jf.setResizable(false);
        jf.setBounds(x1,y1,457,450);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);

        rd=new JLabel("Please wait. Reading data from
glucometer....");
        rd.setVisible(false);
        rd.setBounds(100,100,300,30);
        jf.add(rd);

        patinfo=new JPanel(true);

```

```

        patinfo.setVisible(false);
        patinfo.setLayout(null);
        patinfo.setBounds(0,0,450,400);
        patinfo.setBackground(Color.pink);
        Border raise=BorderFactory.createRaisedBevelBorder();
        Border
        titled=BorderFactory.createTitledBorder(raise,"Register");
        patinfo.setBorder(titled);
        jf.add(patinfo);

        lpname=new JLabel("Patient Name :");
        lpname.setVisible(true);
        lpname.setBounds(50,50,100,40);
        patinfo.add(lpname);

        lpid=new JLabel("Medical ID :");
        lpid.setVisible(true);
        lpid.setBounds(50,125,100,40);
        patinfo.add(lpid);

        lpname=new JLabel("E-Mail ID :");
        lpname.setVisible(true);
        lpname.setBounds(50,200,100,40);
        patinfo.add(lpname);

        pname=new JTextField();
        pname.setVisible(true);
        pname.setBounds(200,60,150,25);
        patinfo.add(pname);

        pid=new JTextField();
        pid.setVisible(true);
        pid.setBounds(200,135,150,25);
        patinfo.add(pid);

        pname=new JTextField();
        pname.setVisible(true);
        pname.setBounds(200,210,150,25);
        patinfo.add(pname);

```

```

submit=new JButton("Register");
submit.setVisible(true);
submit.setBounds(150,300,100,30);
patinfo.add(submit);

submit.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try{
            int id=Integer.parseInt(pid.getText());
            FileOutputStream o=new FileOutputStream("patinfo");
            o.write(id);
            patinfo.setVisible(false);
            rd.setVisible(true);
            new CommPortOpen(null).converse();
        }
        catch(Exception e1){System.out.println(e1);}
    }
});

int i1=450;
int j1=400;
for(;i1>10 && j1>10;i1-=10,j1-=8)
{
    Thread.sleep(4);
    jf.setBounds(x1,y1,i1,j1);

}

for(i=0,j=0;i<450;i+=10,j+=8)
{
    if(j>400)
        j=400;
    Thread.sleep(4);
    jf.setBounds(x1,y1,i,j);
}
jf.setBounds(x1,y1,457,433);

```

```

File f=new File("patinfo");
if(!f.exists())
{
rd.setVisible(false);
    f.createNewFile();

    patinfo.setVisible(true);

}
else
{
rd.setVisible(true);
new CommPortOpen(null).converse();
}

boolean s=true;
x1=screenWidth/2;
y1=screenHeight/2;

}
catch (Exception e)
{}
}
class RemindTask extends TimerTask
{
int i,j;
public RemindTask(int i,int j)
{
    this.i=i;
    this.j=j;
}
public void run()
{
    jf.setBounds(x1,y1,457,433);
}
}

```

```

        public static void main(String ar[])
        {
            proj p=new proj();
        }
    }

```

CommPortOpen.java

```

import java.awt.*;
import java.io.*;
import gnu.io.*;

public class CommPortOpen {

    public static final int TIMEOUTSECONDS = 30;
    public static final int BAUD = 2400;
    protected Frame parent;
    protected DataInputStream is;
    protected PrintStream os;
    protected String response;
    protected boolean debug = true;
    CommPortIdentifier thePortID;
    CommPort thePort;

    public CommPortOpen(Frame f)
    throws IOException, NoSuchPortException, PortInUseException,
    UnsupportedOperationException {

        PortChooser chooser = new PortChooser(null);
        String portName = null;
        do {
            chooser.setVisible(true);
            portName = chooser.getSelectedName();
            if (portName == null) {
                System.out.println("No port selected. Try again.\n");
            }
        } while (portName == null);
        thePortID = chooser.getSelectedIdentifier();
        System.out.println("Trying to open " + thePortID.getName() + "...");
        switch (thePortID.getPortType()) {

```

```

case CommPortIdentifier.PORT_SERIAL:
thePort = thePortID.open("DarwinSys DataComm",
TIMEOUTSECONDS * 1000);
SerialPort myPort = (SerialPort) thePort;
myPort.setSerialPortParams(BAUD, SerialPort.DATABITS_8,
SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
System.out.println("Port Opened");
break;
case CommPortIdentifier.PORT_PARALLEL:
break;
default:
throw new IllegalStateException("Unknown port type " + thePortID);
}
try {
is = new DataInputStream(thePort.getInputStream());
byte[] b = new byte[2];
System.out.println("Reading Data....");
Thread.sleep(5000);
while (true) {
is.read(b);
int val = (b[0] * 0x1000) | b[1];
val=val*-1;
if (val!=0)
{
javax.swing.JOptionPane.showMessageDialog(null,"Your Blood Sugar
Value is 1000");
/new Mailer(""+1000);
}
System.out.println(val);
}
} catch (Exception e) {
System.err.println("Can't open input stream: write-only" + e);
is = null;
}
os = new PrintStream(thePort.getOutputStream(), true);
}

protected void converse() throws IOException {
System.out.println("Ready to read and write port.");
if (is != null) {

```

```

is.close();
}
os.close();
}
}

```

PortChooser.java

```

import gnu.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

public class PortChooser extends JDialog implements ItemListener {

    protected HashMap map = new HashMap();
    protected String selectedPortName;
    protected CommPortIdentifier selectedPortIdentifier;
    protected JComboBox serialPortsChoice;
    protected JComboBox parallelPortsChoice;
    protected JComboBox other;
    protected SerialPort ttya;
    protected JLabel choice;
    protected final int PAD = 5;

    public void itemStateChanged(ItemEvent e) {
        selectedPortName = (String) ((JComboBox)
        e.getSource()).getSelectedItem();
        selectedPortIdentifier = (CommPortIdentifier) map.get(selectedPortName);
        choice.setText(selectedPortName);
    }

    public String getSelectedName() {
        return selectedPortName;
    }

    public CommPortIdentifier getSelectedIdentifier() {
        return selectedPortIdentifier;
    }
}

```



```

public static void main(String[] ap) {
    PortChooser c = new PortChooser(null);
    c.setVisible(true);
    System.out.println("You chose " + c.getSelectedName()
        + " (known by " + c.getSelectedIdentifier() + ").");
    System.exit(0);
}

public PortChooser(JFrame parent) {
    super(parent, "Port Chooser", true);
    makeGUI();
    populate();
    finishGUI();
}

protected void makeGUI() {
    Container cp = getContentPane();
    JPanel centerPanel = new JPanel();
    cp.add(BorderLayout.CENTER, centerPanel);
    centerPanel.setLayout(new GridLayout(0, 2, PAD, PAD));
    centerPanel.add(new JLabel("Serial Ports", JLabel.RIGHT));
    serialPortsChoice = new JComboBox();
    centerPanel.add(serialPortsChoice);
    serialPortsChoice.setEnabled(false);
    centerPanel.add(new JLabel("Your choice:", JLabel.RIGHT));
    centerPanel.add(choice = new JLabel());
    JButton okButton;
    cp.add(BorderLayout.SOUTH, okButton = new JButton("OK"));
    okButton.addActionListener(new ActionListener() {

        public void actionPerformed(ActionEvent e) {
            PortChooser.this.dispose();
        }
    });
}

protected void populate() {
    Enumeration pList = CommPortIdentifier.getPortIdentifiers();
    while (pList.hasMoreElements()) {

```

```

CommPortIdentifier cpi = (CommPortIdentifier) pList.nextElement();
map.put(cpi.getName(), cpi);
if (cpi.getPortType() == CommPortIdentifier.PORT_SERIAL) {
serialPortsChoice.setEnabled(true);
serialPortsChoice.addItem(cpi.getName());
}
}
serialPortsChoice.setSelectedIndex(-1);
}
protected void finishGUI() {
serialPortsChoice.addItemListener(this);
pack();
}
}

```

Mail Retrieval

The doctor side GUI program consists of option to retrieve mail from the client once the reading is received through email. The coding is as follows.

Main.java

```

import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.Flags.Flag;
import javax.mail.search.FlagTerm;
import java.io.*;
import java.sql.*;
public class Main {

Folder inbox;
public Main() {
String id,value,date;
Properties props = System.getProperties();
props.setProperty("mail.store.protocol", "imaps");

try {

Session session = Session.getDefaultInstance(props, null);
Store store = session.getStore("imaps");

```

```

store.connect("imap.gmail.com", "pervasivetech@gmail.com",
"pervasive");

inbox = store.getFolder("Inbox");

int n=inbox.getUnreadMessageCount();
if(n==0)
{
javax.swing.JOptionPane.showMessageDialog(null,"No new mails
Doctor");
}

inbox.open(Folder.READ_WRITE);

Message messages[] = inbox.search(new FlagTerm(new Flags(Flag.SEEN),
false));

FetchProfile fp = new FetchProfile();
fp.add(FetchProfile.Item.ENVELOPE);
fp.add(FetchProfile.Item.CONTENT_INFO);
inbox.fetch(messages, fp);

try {
printAllMessages(messages);
int messn=messages.length;
javax.swing.JOptionPane.showMessageDialog(null,"New mails updated
:"+messn);
inbox.close(true);
store.close();
} catch (Exception ex) {
System.out.println("Exception arise at the time of read mail");
ex.printStackTrace();
}
} catch (NoSuchProviderException e) {
e.printStackTrace();
System.exit(1);
} catch (MessagingException e) {
e.printStackTrace();
}

```

```

System.exit(2);
}
}

```

```

public void printAllMessages(Message[] msgs) throws Exception {
for (int i = 0; i < msgs.length; i++) {
printEnvelope(msgs[i]);
}
}

```

```

public void printEnvelope(Message message) throws Exception {

```

```

String subject = message.getSubject();
java.util.Date receivedDate = message.getReceivedDate();

```

```

getContent(message,subject,receivedDate);
}

```

```

public void getContent(Message msg,String s,java.util.Date d) {
try {

```

```

Multipart mp = (Multipart) msg.getContent();
int count = mp.getCount();
for (int i = 0; i < count-1; i++) {
dumpPart(mp.getBodyPart(i),s,d);
}

```

```

msg.setFlag(Flag.SEEN, true);
} catch (Exception ex) {
System.out.println("Exception arise at get Content");
ex.printStackTrace();
}
}

```

```

public void dumpPart(Part p,String s,java.util.Date da) throws Exception {

```

```

InputStream is = p.getInputStream();
java.io.DataInputStream d=new java.io.DataInputStream(is);

```

```

int c;

```

```

String mess="";
String ch="EOD";

while((mess=d.readLine())!=null)
{
if(!mess.equals(ch))
{
try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:patient","", "");
Statement st=con.createStatement();

String s3="" +da;
String s4[]=s3.split(" ");
String s5=s4[2]+"-"+s4[1]+"-"+s4[5];
javax.swing.JOptionPane.showMessageDialog(null,s+s5+mess);
st.executeUpdate("insert into glucoreading
values('"+s+"','"+s5+"','"+mess+"')");
}
catch(Exception e)
{
javax.swing.JOptionPane.showMessageDialog(null,e);
}
}
else
break;
}
}
}

```

doc.java

```

import javax.swing.*;
import javax.swing.border.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.awt.Toolkit;
import java.util.Timer;

```

```

import javax.swing.table.*;
import java.text.*;
import java.util.TimerTask;
import java.sql.*;

public class doc extends JApplet {

    JFrame auth, docinfo, bchart;
    JTextField user;
    JPanel im, patlist, patbio, bc, pan3, pan4;
    JPasswordField pass;
    JButton log;
    JLabel lpass, luser, pinfo;
    JButton back;
    JTextField pname, pname1, pid, pmid;
    JLabel lpname, lpid, lpmid, lread, lfrom, lto;
    int i, j;
    JTable list, sellist;
    int screenHeight, screenWidth;
    JScrollPane scrollPane;
    JButton tabdata, getchart;
    JButton btnDate;
    JButton btnClearDate;
    JButton btnDate1;
    JButton btnClearDate1;
    JCalendar jCalendar = null;
    JCalendar jCalendar1 = null;
    JDateTextField txtDate;
    JDateTextField txtDate2;
    int count, count1;
    Font fp = new Font("Times New Roman", Font.BOLD, 18);
    Connection con;
    Statement st;
    ResultSet rs;
    int m = 0, cnt;
    String[] columnNames = {"ID", "Name", "Address", "Phone"};
    String[] tableNames = {"Date", "Value"};
    String[][] data;
    String[][] sellistdata;
    int defaultval = 50;

```

```

int vs;
String dname = "Not Assigned";
String[][] seldata;
SimpleBarChart n;
JButton getmail;

public doc() {
try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con = DriverManager.getConnection("jdbc:odbc:patient", "", "");
st = con.createStatement();

GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
GraphicsDevice[] gs = ge.getScreenDevices();
DisplayMode dm = gs[0].getDisplayMode();
screenWidth = dm.getWidth();
screenHeight = dm.getHeight();
i = screenWidth;
j = screenHeight;

auth = new JFrame("Log In");
auth.setLayout(null);
auth.setVisible(true);
auth.setBounds((i / 3), (j / 3), 351, 351);
auth.setResizable(false);
auth.setDefaultCloseOperation(auth.EXIT_ON_CLOSE);

docinfo = new JFrame("Welcome Doctor");
docinfo.setLayout(null);
docinfo.setVisible(false);
docinfo.setBounds(0, 0, i - 1, (j / 2) + 50);
docinfo.setResizable(true);
docinfo.setDefaultCloseOperation(auth.EXIT_ON_CLOSE);

getmail = new JButton("Get Mail");
getmail.setVisible(true);
getmail.setBounds((i / 2) - 50, (j / 2), 100, 30);

```

```
docinfo.add(getmail);
```

```
patbio = new JPanel(true);
patbio.setVisible(false);
patbio.setLayout(null);
patbio.setBounds((i / 2), 0, (i), (j / 2));
patbio.setBackground(Color.pink);
Border raise6 = BorderFactory.createRaisedBevelBorder();
Border titled6 = BorderFactory.createTitledBorder(raise6, "PATIENT
BIODATA");
patbio.setBorder(titled6);
docinfo.add(patbio);
```

```
patlist = new JPanel(true);
patlist.setVisible(false);
patlist.setLayout(new BorderLayout());
patlist.setBounds(0, 0, (i / 2), (j / 2));
patlist.setBackground(Color.pink);
Border raise1 = BorderFactory.createRaisedBevelBorder();
Border titled1 = BorderFactory.createTitledBorder(raise1, "PATIENT
LIST");
patlist.setBorder(titled1);
docinfo.add(patlist);
```

```
lpname = new JLabel("Patient Name :");
lpname.setVisible(true);
lpname.setBounds(50, 50, 100, 40);
patbio.add(lpname);
```

```
lpid = new JLabel("Medical ID :");
lpid.setVisible(true);
lpid.setBounds(50, 125, 100, 40);
patbio.add(lpid);
```

```
lpname = new JLabel("Phone No :");
lpname.setVisible(true);
lpname.setBounds(50, 200, 100, 40);
patbio.add(lpname);
```



```
pname = new JTextField();  
pname.setVisible(true);  
pname.setBounds(200, 60, 150, 25);  
pname.setEditable(false);  
patbio.add(pname);
```

```
pid = new JTextField();  
pid.setVisible(true);  
pid.setBounds(200, 135, 150, 25);  
pid.setEditable(false);  
patbio.add(pid);
```

```
pname1 = new JTextField();  
pname1.setVisible(true);  
pname1.setBounds(200, 210, 150, 25);  
pname1.setEditable(false);  
patbio.add(pname1);
```

```
lread = new JLabel("READINGS");  
lread.setVisible(true);  
lread.setBounds(200, 225, 100, 40);  
patbio.add(lread);
```

```
lfrom = new JLabel("From :");  
lfrom.setVisible(true);  
lfrom.setBounds(50, 250, 100, 40);  
patbio.add(lfrom);
```

```
pinfo = new JLabel("PATIENT INFORMATION");  
pinfo.setVisible(true);  
pinfo.setBounds(200, 10, 200, 40);  
patbio.add(pinfo);
```

```
lto = new JLabel("To :");  
lto.setVisible(true);  
lto.setBounds(50, 300, 100, 40);  
patbio.add(lto);
```

```
txtDate = new JDateTextField(JDateTextField.LONG_DATE);
txtDate.setBounds(200, 260, 105, 20);
txtDate.setEditable(false);
txtDate.setVisible(true);
txtDate.setFont(fp);
patbio.add(txtDate);
```

```
btnDate = new JButton(new ImageIcon("date.gif"));
btnDate.setBounds(310, 260, 20, 20);
btnDate.setVisible(true);
patbio.add(btnDate);
```

```
btnClearDate = new JButton(new ImageIcon("delete.gif"));
btnClearDate.setBounds(335, 260, 20, 20);
btnClearDate.setVisible(true);
patbio.add(btnClearDate);
```

```
txtDate2 = new JDateTextField(JDateTextField.LONG_DATE);
txtDate2.setBounds(200, 310, 105, 20);
txtDate2.setEditable(false);
txtDate2.setVisible(true);
txtDate2.setFont(fp);
patbio.add(txtDate2);
```

```
btnDate1 = new JButton(new ImageIcon("date.gif"));
btnDate1.setBounds(310, 310, 20, 20);
btnDate1.setVisible(true);
patbio.add(btnDate1);
```

```
btnClearDate1 = new JButton(new ImageIcon("delete.gif"));
btnClearDate1.setBounds(335, 310, 20, 20);
btnClearDate1.setVisible(true);
patbio.add(btnClearDate1);
```

```
getchart = new JButton("Get Chart");
getchart.setVisible(true);
getchart.setBounds(200, 345, 100, 30);
patbio.add(getchart);
```

```
getchart.addActionListener(new ActionListener() {
```

```

public void actionPerformed(ActionEvent e) {
    try {
        docinfo.setVisible(false);

        bchart = new JFrame("Patient Status");
        bchart.setLayout(null);
        bchart.setVisible(true);
        bchart.setBounds(200, 200, 800, 800);
        bchart.setResizable(false);
        bchart.setDefaultCloseOperation(auth.EXIT_ON_CLOSE);

        bc = new JPanel(true);
        bc.setVisible(true);
        bc.setLayout(new BorderLayout());
        bc.setBounds(400, 0, 400, 400);
        bc.setBackground(Color.pink);
        Border raise2 = BorderFactory.createRaisedBevelBorder();
        Border titled2 = BorderFactory.createTitledBorder(raise2, "PATIENT
        READINGS");
        bc.setBorder(titled2);
        bchart.add(bc);

        back = new JButton("Done");
        back.setVisible(true);
        back.setBounds(10, 420, 100, 30);
        bchart.add(back);

        int count = 0;
        int k = 0;
        String id = pid.getText();
        seldata = new String[10][2];

        rs = st.executeQuery("select rdate,value from glucoreading where ID='" + id
        + "' and rdate between #" + txtDate.getText(13) + "# and #" +
        txtDate2.getText(13) + "# order by rdate desc");
        if (k >= 5) {
            while (rs.next()) {
                seldata[count][0] = "" + rs.getDate(1);
            }
        }
    }
}

```

```

seldata[count][1] = rs.getString(2);
count++;
}
} else {
while (rs.next()) {
seldata[count][0] = "" + rs.getDate(1);
seldata[count][1] = rs.getString(2);
count++;
}
while (count <= 5) {
seldata[count][0] = dname;
seldata[count][1] = "" + defaultval;
count++;
}
}
rs.close();

```

```

double[] values = new double[5];
String[] names = new String[5];
for (vs = 0; vs < 5; vs++) {
names[vs] = seldata[vs][0];
values[vs] = Double.parseDouble(seldata[vs][1]);
}

```

```

SimpleBarChart n = new SimpleBarChart(values, names, "Names");
n.setBounds(0, 0, 400, 400);
n.setBackground(Color.pink);
bchart.add(n);
bchart.setBounds(200, 200, 810, 501);

```

```

m = 0;
rs = st.executeQuery("select count(*) from glucoreading where ID=" + id +
""");
if (rs.next()) {
cnt = rs.getInt(1);
}
rs.close();

```

```

rs = st.executeQuery("select rdate,value from glucoreading where ID='" + id
+ "'");
sellistdata = new String[cnt][2];
while (rs.next()) {
sellistdata[m][0] = "" + rs.getDate(1);
sellistdata[m][1] = rs.getString(2);
m++;
}
m = 0;
rs.close();

```

```

sellist = new JTable(sellistdata, tableNames);
bc.add(sellist, BorderLayout.CENTER);
bc.add(sellist.getTableHeader(), BorderLayout.NORTH);
sellist.setVisible(true);
sellist.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
bchart.add(bc);

```

```

back.addActionListener(new ActionListener() {

```

```

    public void actionPerformed(ActionEvent e) {
        bchart.setVisible(false);
        docinfo.setVisible(true);
    }
});

```

```

    } catch (Exception e2) {
        bchart.dispose();
        docinfo.show();
        JOptionPane.showMessageDialog(null, "Please select the date fields");
    }
    }
});

```

```

count = 0;

```

```

jCalendar = new JCalendar(docinfo, "Calendar", true);
btnDate.addActionListener(new ActionListener() {

```

```

public void actionPerformed(ActionEvent e) {
    Calendar cDate = jCalendar.getCalendar();
    if (cDate == null) {
        cDate = Calendar.getInstance();
    }
    jCalendar.setCalendar(cDate);
    jCalendar.setVisible(true);

    if (jCalendar.isOkPressed()) {
        txtDate.setDate(jCalendar.getDate());
        count = 1;
    }

    jCalendar.setVisible(false);
}

btnClearDate.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        txtDate.setDate((java.util.Date) null);
        jCalendar.setDate(null);
        count = 0;
    }
});

count1 = 0;

jCalendar1 = new JCalendar(docinfo, "Calendar", true);
btnDate1.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        Calendar cDate = jCalendar.getCalendar();
        if (cDate == null) {
            cDate = Calendar.getInstance();
        }
        jCalendar.setCalendar(cDate);
        jCalendar.setVisible(true);
    }
});

```

```

if (jCalendar.isOkPressed()) {
    txtDate2.setDate(jCalendar.getDate());
    count = 1;
}

jCalendar.setVisible(false);
}
});

btnClearDate1.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        txtDate2.setDate((java.util.Date) null);
        jCalendar.setDate(null);
        count = 0;
    }
});

rs = st.executeQuery("select count(*) from patient");
if (rs.next()) {
    cnt = rs.getInt(1);
}
rs.close();
rs = st.executeQuery("select * from patient");
data = new String[cnt][4];
while (rs.next()) {
    data[m][0] = rs.getString(1);
    data[m][1] = rs.getString(2);
    data[m][2] = rs.getString(3);
    data[m][3] = rs.getString(4);
    m++;
}
m = 0;
rs.close();

list = new JTable(data, columnNames);
patlist.add(list, BorderLayout.CENTER);
patlist.add(list.getTableHeader(), BorderLayout.NORTH);

```

```

list.setVisible(true);
list.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
patlist.add(list);

tabdata = new JButton("Get Info");
tabdata.setVisible(false);
patlist.add(tabdata, BorderLayout.SOUTH);

tabdata.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        try {
            int sel = list.getSelectedRow();
            String[] da = new String[4];
            da[0] = data[sel][0];
            rs = st.executeQuery("select * from patient where ID='" + data[sel][0] + "'");
            if (rs.next()) {
                pid.setText(rs.getString("ID"));
                pname.setText(rs.getString("Name"));
                pname1.setText(rs.getString("Phone"));
            }
        } catch (Exception e1) {
            JOptionPane.showMessageDialog(null, "Please select a Patient Name and
            get his information");
        }
    }
});

luser = new JLabel("UserName :");
luser.setVisible(true);
luser.setBounds(50, 50, 100, 40);
auth.add(luser);

lpass = new JLabel("Password :");
lpass.setVisible(true);
lpass.setBounds(50, 125, 100, 40);
auth.add(lpass);

user = new JTextField();

```



```
user.setVisible(true);
user.setBounds(150, 60, 150, 25);
auth.add(user);
```

```
pass = new JPasswordField();
pass.setVisible(true);
pass.setBounds(150, 135, 150, 25);
auth.add(pass);
```

```
log = new JButton("Log In");
log.setVisible(true);
log.setBounds(125, 225, 100, 30);
auth.add(log);
```

```
log.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent e) {
        String u = user.getText();
        String p = pass.getText();
        String pa = null;
```

```
        try {
            if ((!u.equals("")) && (!p.equals(""))) {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con = DriverManager.getConnection("jdbc:odbc:patient", "", "");
                st = con.createStatement();
                rs = st.executeQuery("select pass from admin where user='" + u + "'");
                if (rs.next()) {
                    pa = rs.getString(1);
                }
                if (p.equals(pa)) {
                    auth.dispose();
                    docinfo.setVisible(true);
                    patbio.setVisible(true);
                    patlist.setVisible(true);
                    docinfo.setBounds(0, 0, i, (j / 2) + 70);
                    list.setVisible(true);
                    tabdata.setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid Authentication");
                }
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
    }
});
```

```

auth.dispose();
System.exit(0);
}
} else {
JOptionPane.showMessageDialog(null, "Please Enter all the fields");
user.setText("");
pass.setText("");
}
} catch (Exception u1) {
JOptionPane.showMessageDialog(null, u1);
}

}
});

auth.setBounds((i / 3), (j / 3), 350, 350);

} catch (Exception e) {
JOptionPane.showMessageDialog(null, e);
}
getmail.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {
new Main();
}
});
}
Public static void main(String ar[])
{
    Doc d=new doc();
}
}

```

SimpleBarChart.java

```

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class SimpleBarChart extends JPanel {

```

```

private double[] value;
private String[] languages;
private String title;

public SimpleBarChart(double[] val, String[] lang, String t) {
    languages = lang;
    value = val;
    title = t;
}

public void paintComponent(Graphics graphics) {
    super.paintComponent(graphics);
    if (value == null || value.length == 0)
        return;
    double minVal = 0;
    double maxVal = 0;
    for (int i = 0; i < value.length; i++) {
        if (minVal > value[i])
            minVal = value[i];
        if (maxVal < value[i])
            maxVal = value[i];
    }
    Dimension dim = getSize();
    int clientWidth = dim.width;
    int clientHeight = dim.height;
    int barWidth = clientWidth / value.length;
    Font titleFont = new Font("Book Antiqua", Font.BOLD, 15);
    FontMetrics titleFontMetrics = graphics.getFontMetrics(titleFont);
    Font labelFont = new Font("Book Antiqua", Font.PLAIN, 10);
    FontMetrics labelFontMetrics = graphics.getFontMetrics(labelFont);
    int titleWidth = titleFontMetrics.stringWidth(title);
    int q = titleFontMetrics.getAscent();
    int p = (clientWidth - titleWidth) / 2;
    graphics.setFont(titleFont);
    graphics.drawString(title, p, q);
    int top = titleFontMetrics.getHeight();
    int bottom = labelFontMetrics.getHeight();
    if (maxVal == minVal)
        return;
    double scale = (clientHeight - top - bottom) / (maxVal - minVal);
    q = clientHeight - labelFontMetrics.getDescent();

```

```

graphics.setFont(labelFont);

for (int j = 0; j < value.length; j++) {
    int valueP = j * barWidth + 1;
    int valueQ = top;
    int height = (int) (value[j] * scale);
    if (value[j] >= 0)
        valueQ += (int) ((maxValue - value[j]) * scale);
    else {
        valueQ += (int) (maxValue * scale);
        height = -height;
    }

    graphics.setColor(Color.blue);
    graphics.fillRect(valueP, valueQ, barWidth - 2, height);
    graphics.setColor(Color.black);
    graphics.drawRect(valueP, valueQ, barWidth - 2, height);
    int labelWidth = labelFontMetrics.stringWidth(languages[j]);
    p = j * barWidth + (barWidth - labelWidth) / 2;
    graphics.drawString(languages[j], p, q);
}
}

```