REMOTE MEDICAL INC.

School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6
ensc440-rabbit@sfu.ca

March 11th, 2005

Lakshman One
School of Engineering Science
Simon Fraser University
Burnaby BC
V5A 1S6

RE: ENSC 440 Project Remote Health Monitor Design Specification

Dear Lucky,

We have attached the Remote Health Monitor Design Specification based on the project described in the proposal we submitted previously. Remote Health Monitor is to collect a patient's health data and manage them in a low cost, scaleable, and user-friendly fashion. Patients can be monitored anywhere there is the internet access. The Remote Health Monitor measures health parameters such as body temperature and pulse, and then transmitted the readings to the hospital's database through the internet. With the software that comes with the package installed, patients can also access their personal clinical information.

The design specification describes in detail the implementations of each module of the remote health monitoring system. The operations of each component of our system, as well as overall system description, are discussed.

Remote Medical Inc. consists of four experienced and enthusiastic 3rd and 4th year engineering students: Dong Zhang, Calvin Che, Marian Chang, and Lotus Yi. If you have any questions or concerns, please feel free to contact us through the email at ensc440-rabbit@sfu.ca.

Sincerely,

Lotus Yi

Lotus Yi
Chief Executive Officer
Remote Medical Inc.

Enclosure: *Design Specification for Remote Health Monitor*

REMOTE MEDICAL INC.

Design Specification for
Remote Health Monitor

Project Team:

Marian Chang
Calvin Che
Lotus Yi
Dong Zhang

Contact Person:

Lotus Yi
ensc440-rabbit@sfu.ca

Submitted to:

Lucky One – ENSC 440
Mike Sjoerdsma – ENSC 305
School of Engineering
Simon Fraser University

Issued Date:

March 11, 2005

ABSTRACT

Management of patients with chronic conditions is a long-standing challenge for health care organizations. These conditions include diabetes, chronic heart failure, asthma, HIV / AIDS, and cancer. Patients are required to adopt lifelong diet and drug control to maintain optimal health and avoid the complications of the disease. These complications can arise suddenly and can be life threatening. Therefore, the health condition of patients with chronic diseases should be reported frequently.

We propose to implement the data reporting into a single stationary device that transmits health information for physicians via internet connections. Our solution, Remote Health Monitor (RHM), measures a patient's medical conditions and sends the measurements to the hospital database through internet. The design of a stationary device ensures consistency of the environment under which measurements are taken. The database can be accessed either by the physicians or patients with specific login identifications to keep information confidential.

This document, the design specification, describes in detail the implementation of our health monitoring system. Hardware mechanisms used to obtain the health data, firmware methods used to process the health data, and the software algorithms used to manage the health data are discussed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

API	Application Program Interface, the interface application programs use for accessing services provided by some lower-level module
bits	Fundamental of digital information with possible values of 0 and 1.
bytes	A collection of 8 bits
EEPROM	Electrically Erasable Programmable Read-Only Memory, the memory that store the operating system and user programs for a chip, and can be erased and reprogrammed with an external device. It saves data even when power is loss.
interrupt	The suspension of normal program execution to perform a higher priority task as requested by a peripheral device. After completion of the task, the execution of the interrupted program is resumes from the point where it was left off.
I/O	Input and output ports, which is used along with connection cable to link one device to the other and allow communication.
IP	Internet protocol, the basic network transmission protocol of the Internet.
IP address	An address with 4 numbers separate by periods that uniquely identifies a certain computer on the internet.
ISR	Interrupt service routine, a piece of code that the processor executes when an external event, such as a timer, occurs.
key	The column in a database table.
LED	Light-emitting diode, a semiconductor device that produces light when conducting currents.
MySQL	An open-source SQL database developed at www.mysql.com. It is common used for small business.
opcode	Operation code, computer instruction word that designates the function performed by a specific instruction, for example, “ADD” and “SUB”.
primary key	A column that uniquely defines a row in a database table.
protocol	Standard procedure for regulating data transmission.

register	A memory device that has a specific address and is used to hold temporary data.
registry	A database used by Microsoft Windows to store configuration information.
RHM	Remote Health Monitor, the name of our product.
Rabbit 3000	An 8-bit microprocessor that is made by Rabbit Semiconductor Inc. and is commonly used in embedded systems.
SPI	Serial peripheral interface, a method of communication between host processor and peripheral device with data sending one bit at a time through a single channel.
SQL	Structured Query Language, a standard interactive and programming language for getting information from and updating a database.
TCP / IP	Transmission control protocol / internet protocol, a protocol that computer used to communication and exchange information over the internet.

1. INTRODUCTION

The goal of the Remote Health Monitor (RHM) is to collect a patient's health data and manage them. Patients can be monitored from anywhere there is internet access. The RHM measures parameters such as temperature and pulse, and transmits the information to the database located in the hospital. The database can be accessed through a software application that will be provided along with the purchase of RHM. The application can be logged in to either as a doctor or as a patient, thus allowing the care providers to monitor a patient's health condition and the patients to review their own medical history.

Our goal in implementing RHM is to provide chronic disease patients with an easy to use device that will help them track health information without going to the hospital, and to allow doctors to review the patient data with a software application that organizes and updates information automatically.

This document describes in detail the implementation of the remote health monitoring system. It provides high level design overview and the specific features and tasks of each module. Hardware mechanisms used to obtain the health data, firmware methods used to process the health data, and the software algorithms used to manage the health data are discussed and explained. Hardware circuitries and software flowcharts are also provided.

2. SYSTEM OVERVIEW

Figure 1 shows the high level structure of the system. The database server is located in the hospital; the Microcontroller – Rabbit 3000, and measuring devices are installed in the patient's house; the user can access from any PC with the client software installed and internet access, and can log in either in doctor or patient mode.

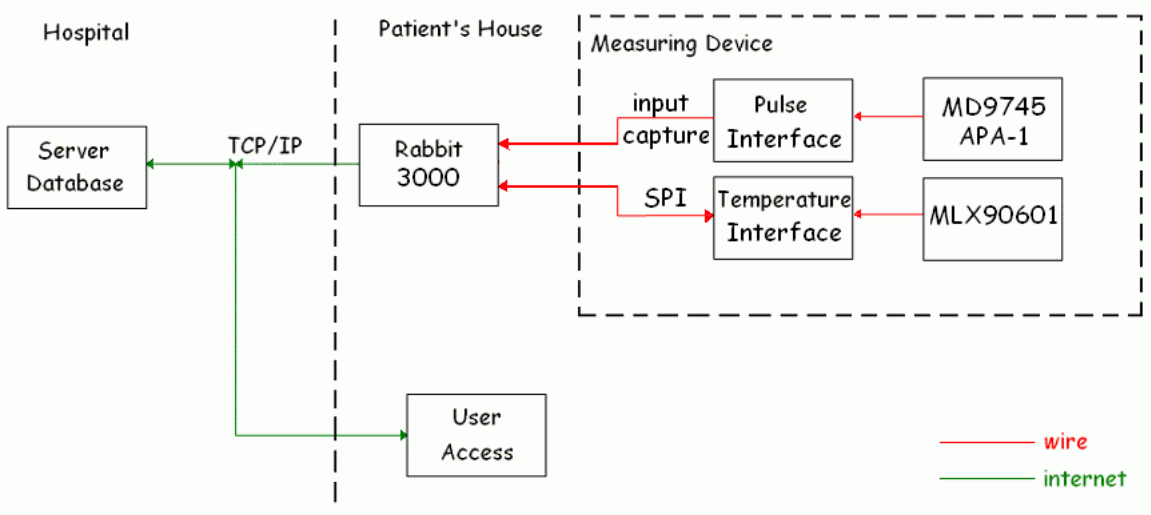

Figure 1: System Block Diagram

Figure 2 presents the basic functionality of RHM in a graphical manner.

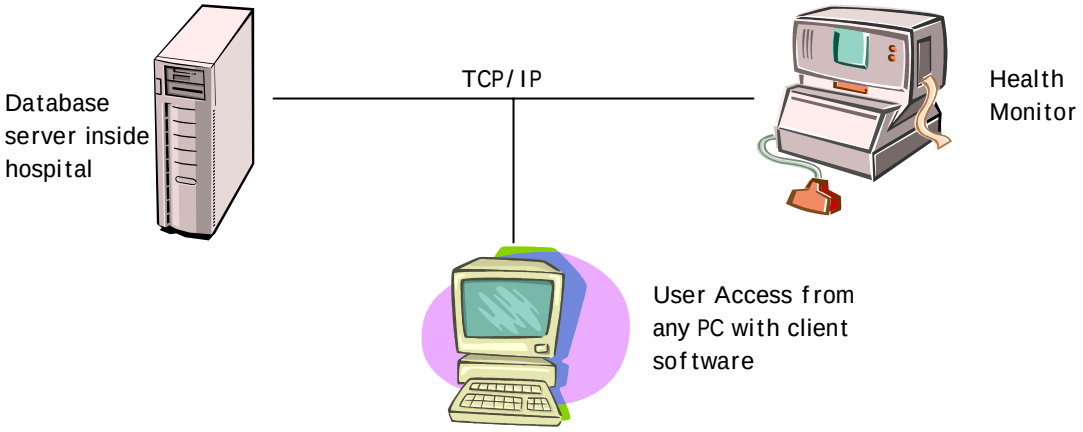

Figure 2: Graphical Diagram of the System

3. DESIGN OF THE FIRMWARE

3.1 User Interface Handler

User interface handler is used to efficiently and correctly obtain user's health data. The figure shown below is the interface board connected to the I/O of the Rabbit 3000. It consists of four push buttons, four LEDs, and a buzzer. The first set of push button (SW1) and LED (LED1) corresponds to pulse measurements, and the second set of push button and LED corresponds to temperature measurements. After the user set up the monitoring device, the push button will be pressed to indicate the start of measurements. If the data obtained is not within a reasonable range, the corresponding LED would flash to indicate a non-successful data acquirement, and the user should restart the measurement by pressing the push button again. User can stop the measurement by pressing the same push button.

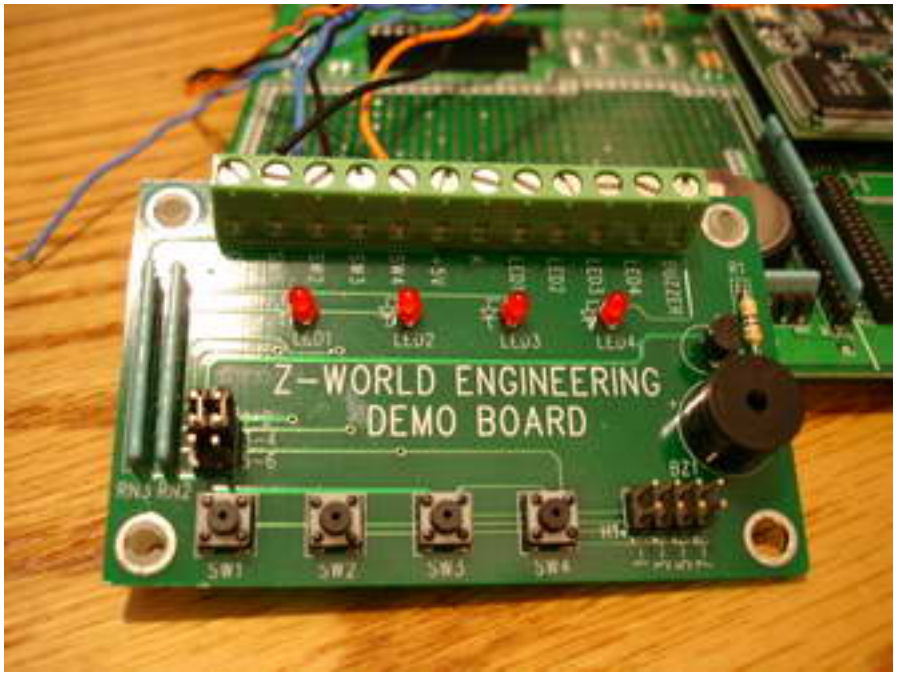

Figure 3: The Demo Board of Rabbit 3000

User interface handler is implemented using the two external interrupts, one for pulse measurement and one for temperature measurement. Two similar interrupt service routines are designed for each of the measurement, and the flow chart of them are shown in Figure 4.

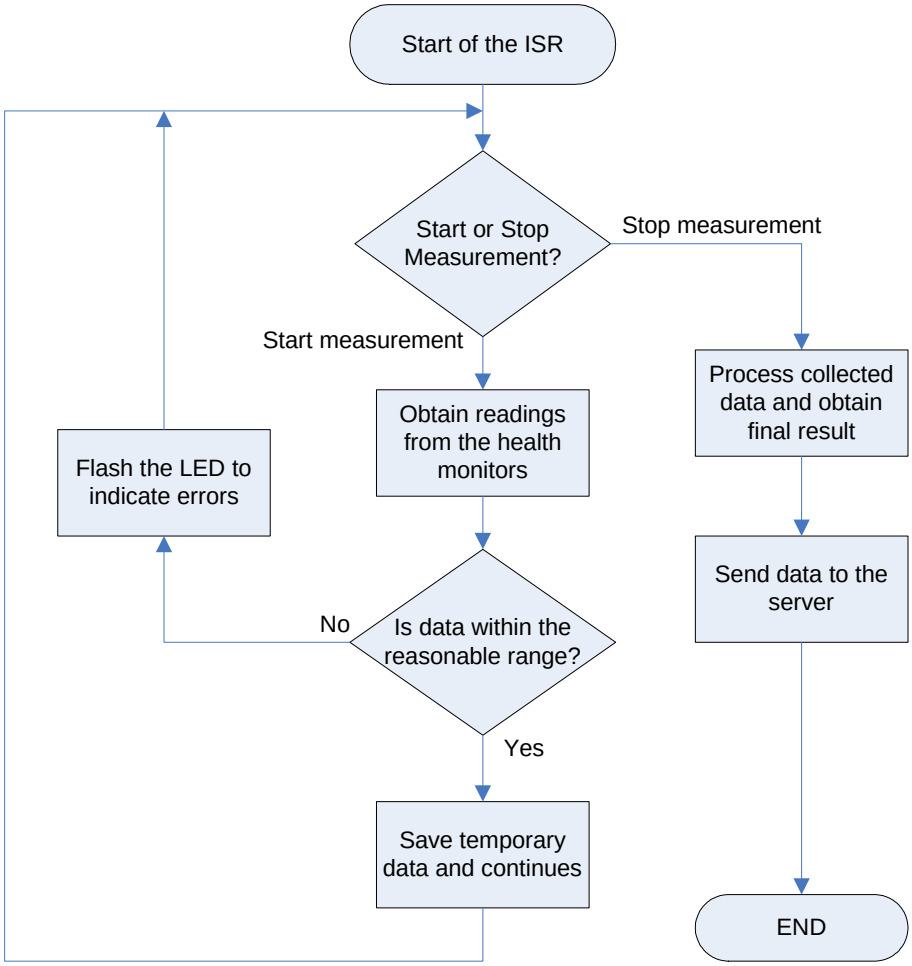

Figure 4: User Interface Handler Flowchart

3.2 Temperature Data Acquirement

3.2.1 SPI Protocol

To obtain the temperature reading, we utilize the serial communication protocol, SPI, to communicate with the infrared thermometer module. The thermometer module has a built-in digital interface that is SPI compatible, and will always work as a slave device. It can be used to access the on-chip EEPROM and all internal registers. The format of any command is always 32 bits: 8 bits for the operation code, 8 bits for the address and 16 bits of data. Every write command starts with a high to low transition of CS (chip select) and ends by a low to high transition of CS after 32 periods of the serial data clock (SCLK). The thermometer module reads the data present on pin SDI (serial data in) on the rising edge of the clock. With a delay of 8 periods of the serial clock, the SPI will repeat the opcode, address and the first 8 bits of data on pin SDO (serial data out). This allows the microcontroller, Rabbit 3000, to check command and address, and terminate the

operation in case of an error by forcing CS high before the end of the complete command cycle, i.e. before the end of the 32 clock periods. The read command is build up similarly, except that no data has to be passed. On SDO, the opcode will be followed directly by the requested data, the address is not returned in this case. The data on SDO is valid on the rising edge of the clock. In case of a read command, the SPI will output the data on SDO starting on the 25th rising edge of the clock (after CS is pulled low).

3.2.2 Timing/Speed

The timing requirement of the thermometer module is illustrated in Figure 5 and Table 1.

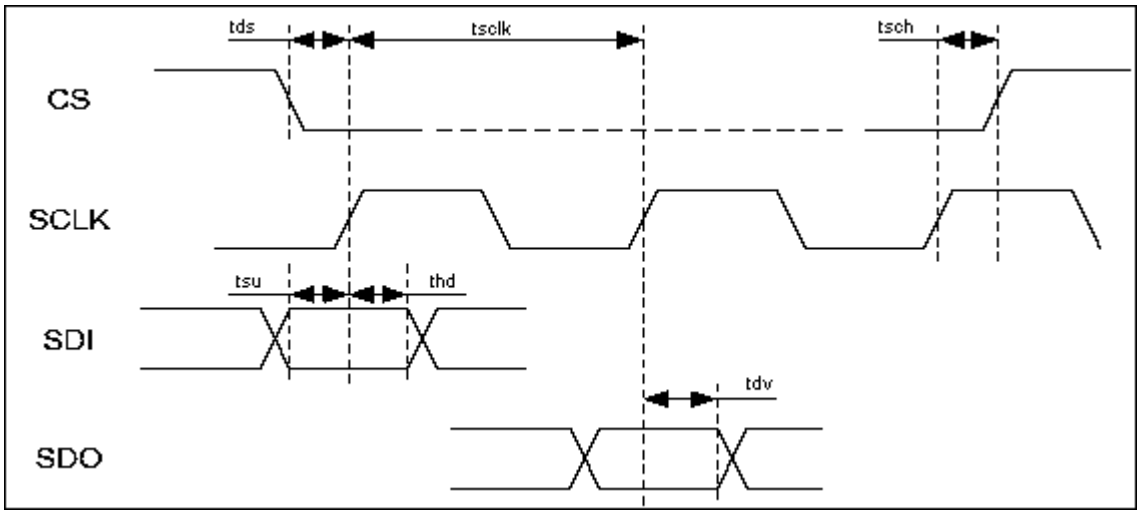

Figure 5: Timing Diagram for SPI

Table 1: Timing Requirement for SPI

Symbol	Parameter	Value	Unit
tsclk	Sclk period	min 8	µs
tcls	CS low to SCLK high	min 50	ns
tsch	SCLK low to CS high	min 50	ns
tsu	Data in setup time	min 200	ns
thd	Data in hold time	min 200	ns
tdv	Data out valid	min 1	µs

3.2.3 Operation Codes and Internal Registers

A list of the operation codes and addresses of internal registers are shown in Table 2 and 3 respectively.

Table 2: Operation Codes

Symbol	Parameter	Command
WR	X101X0XX	Write internal register
RD	X10010XX	Read internal register
WEPR	0001XXXX	Write EEPROM
ER	001XXXXX	Erase EEPROM
REPR	X0001XXX	Read EEPROM
BLWR	1001XXXX	Block write EEPROM
BLER	101XXXXX	Block erase EEPROM

Table 3: Address of Internal Registers

Register	Function	Address
Irout	Tobject (lin)	09h
Tout	Tambint (lin)	0Ah

3.2.4 Implementation Using Rabbit 3000

To implement the SPI interface between the thermometer module and Rabbit 3000, we use the I/O pins PD2 (port D, pin 2) for CS, PB0 (port B, pin 0) for serial clock (75 kHz), PC4 (port C, pin 4) for MOSI (master out, serial in), and PC5 (port C, pin 5) for MISO (master in, serial out). After the start measurement button is pressed, CS is pulled low and required command and clock signals are sent to the thermometer module. CS is pulled high after the data is successfully obtained and saved into the register of Rabbit 3000. The flow chart of this implementation is shown in Figure 5.

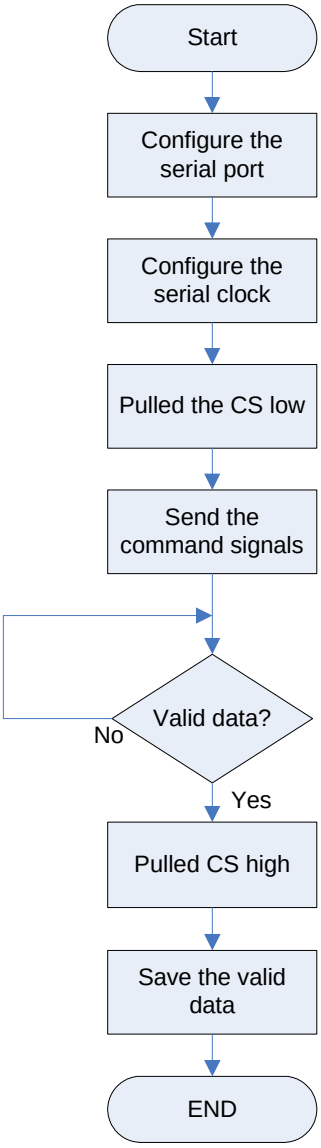

Figure 6: Flowchart for SPI

3.3 Pulse Measurement

For the pulse measurement, we utilize the on-chip input capture channel, which simplifies the solution to software only. Basically, after proper setup, the input capture channel will wait for certain action that is happening on an input pin. There are two conditions defined for the use for the input capture channel: starting condition and ending condition. When the starting condition occurs, the timer is started; after the ending condition is caught by the processor, the time difference between start and end is saved in one register. For our application, since we are measuring the pulse rate, the duration of the square pulse will give us the duration of the pulse, provided that the pulse sensor will produce a square pulse in accordance to the patient's heart beat, namely,

a rising edge for one beat of heart. In our situation, the starting condition and ending condition should both be the rising edge of one pulse. Thus, the time difference will give us the duration for the pulse, from that the heart beat rate can be calculated.

There is one difficulty introduced due to the overflow of the timer. Every time the timer overflows, a roll-over condition is generated to signal this fact, and the client program has to catch this condition to perform appropriate processing to make sure the accuracy of the measurement. To tackle this problem, we setup a variable to keep track of how many times the timer counter has rolled over, and then we add the duration corresponding to the timer required for the roll-overs to track the duration exactly.

Figure 7 below illustrates an example for calculating the duration when roll-over occurs, and the flow chart of the implementation is shown in Figure 8.

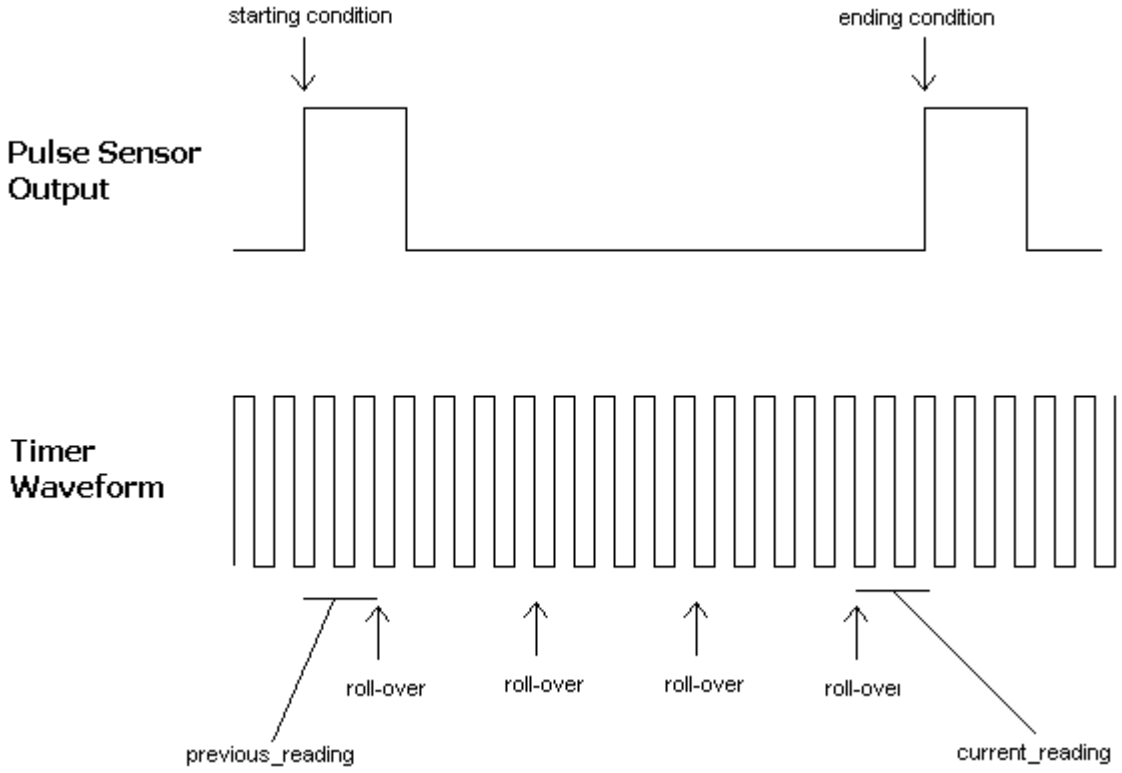

Figure 7: Calculating Roll-Over Duration

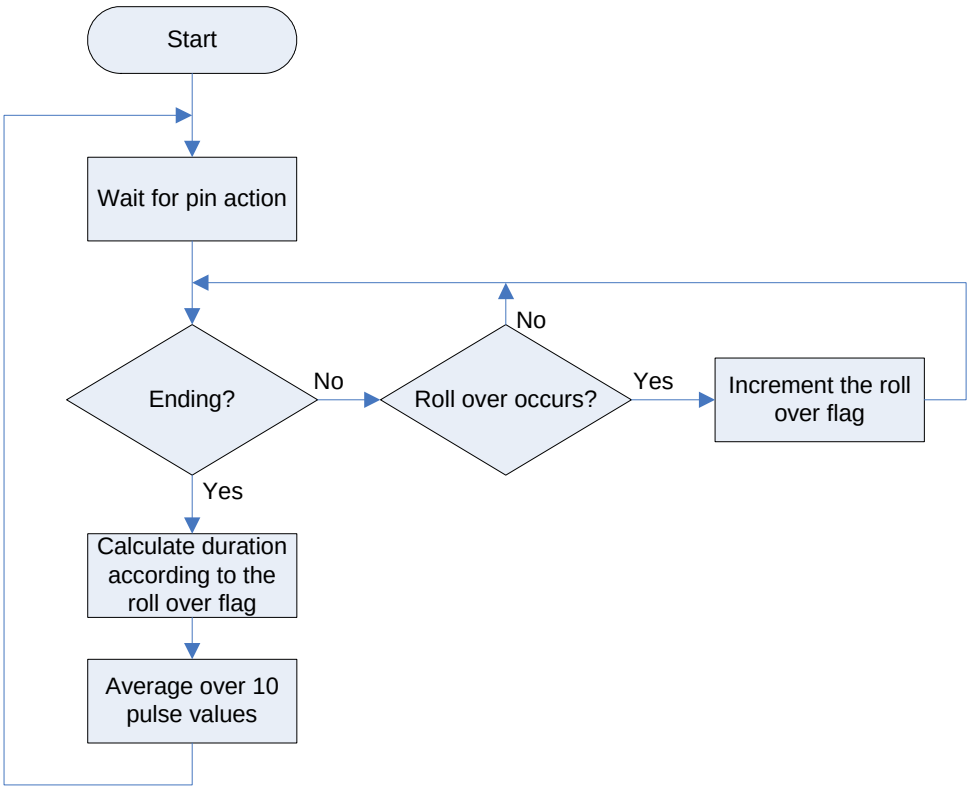

Figure 8: Pulse Measurement Flowchart

3.4 Data Transmission using TCP/IP

3.4.1 TCP Connection

To use reliable transport services, TCP hosts must establish a connection-oriented session with one another. Connection establishment is performed by using a "three-way handshake" mechanism. A three-way handshake synchronizes both ends of a connection by allowing both sides to agree upon initial sequence numbers. This mechanism also guarantees that both sides are ready to transmit data and know that the other side is ready to transmit as well. This is necessary so that packets are not transmitted or retransmitted during session establishment or after session termination. Each host randomly chooses a sequence number used to track bytes within the stream it is sending and receiving. Then, the three-way handshake proceeds in the following manner:

- The first host (Host A) initiates a connection by sending a packet with the initial sequence number (X) and SYN (synchronous) bit set to indicate a connection request.
- The second host (Host B) receives the SYN, records the sequence number X, and replies by acknowledging (ACK) the SYN (with an ACK = X + 1). An ACK = 20 means the host has received bytes 0 through 19 and expects byte 20 next.
- Host B includes its own initial sequence number (SEQ = Y). This technique is called forward acknowledgment.

- Host A then acknowledges all bytes Host B sent with a forward acknowledgment indicating the next byte Host A expects to receive (ACK = Y + 1). Data transfer then can begin.

3.4.2 Implementation Using Rabbit 3000 - Client

For our project, we use the server-client mode to implement the data transmission through the internet using TCP/IP. After the measurement is done, Rabbit 3000, the microcontroller, will initiate a connection to the Database Server and start the transferring of data. The flowchart of the implementation is shown below.

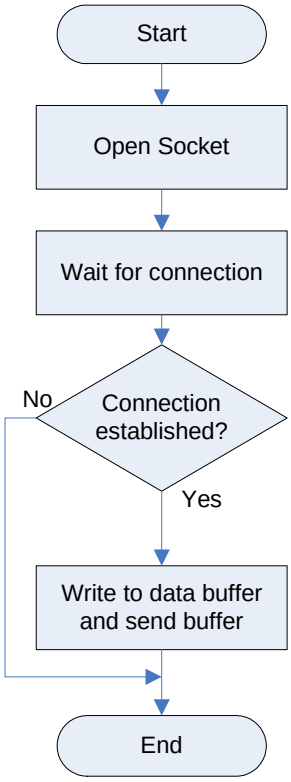

Figure 9: Flowchart for Sending Data between Microcontroller and Database Server

3.4.3 Implementation Using C# - Server

On the Database end, we implement a server written in C# to receive the data sent by Rabbit 3000 and store the data into the database. The flowchart of the implementation is shown below.

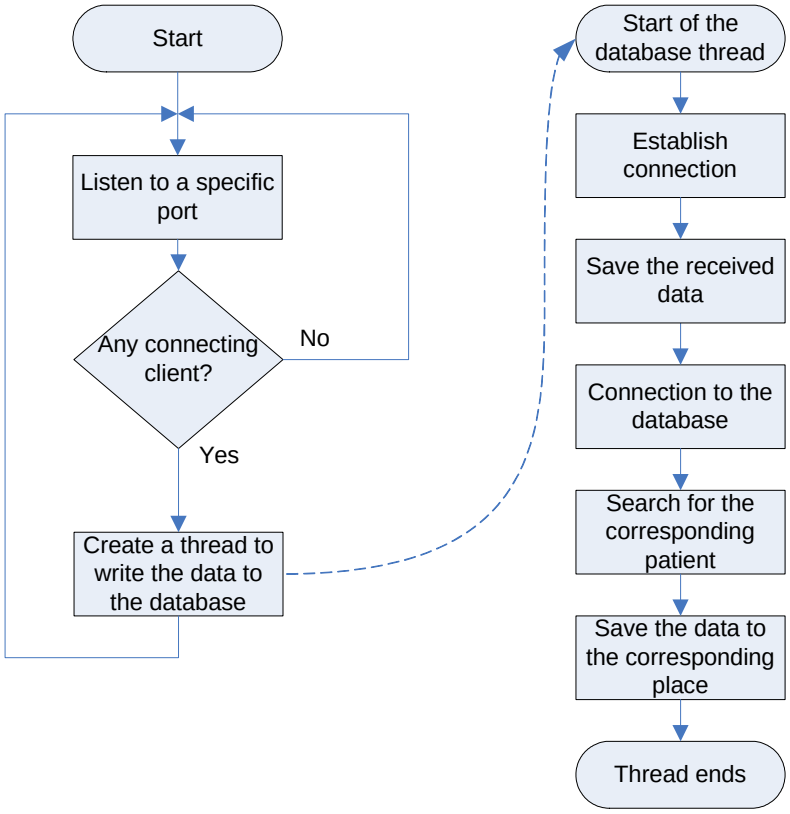

Figure 10: Server Flowchart

4. DESIGN OF PULSE SENSOR

4.1 Architecture Overview

The pulse measurement device consists of a microphone, signal acquisition circuit, and signal detection and amplification circuit. The signal from the device is to be fed into the input capture of our microcontroller for the calculation of the patient's heart beats per minute. The flow of the signal is shown in the following block diagram.

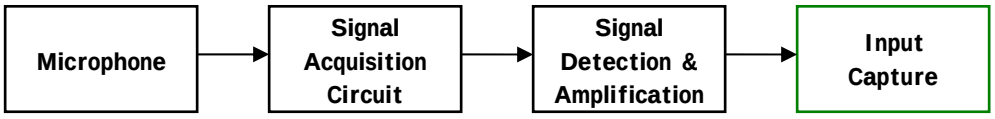

Figure 11: Pulse Measurement Block Diagram

4.2 Microphone

After exploring different possible methods for pulse measurement, including utilizing reflective sensor or pressure sensor, we found using microphone for obtaining the heart beat is most feasible and cost effective. The instrument we have chosen for the measurement of pulse rate is an electret condenser microphone, MD9745APA-1. The microphone will be placed on the patient's chest to sense the heart pumping sound. The pin assignment and their location on the microphone are shown in Figure 12.

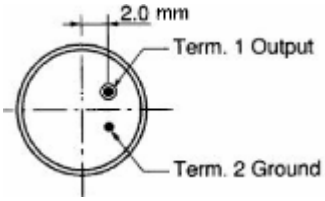

Figure 12: Electret Condenser Microphone Structure

Both pins are connected to the signal acquisition circuit through wiring with length of 1 meter to ensure its mobility when patients attempt to place the microphone on their chests.

4.3 Signal Acquisition

To interpret the signal obtained through the electret condenser microphone, we use the circuit shown in Figure 13.

A resistor of 10kΩ is used to limit the amount of current entering the microphone, which has a maximum current rating of 0.6mA. The output signal acquired would stay at ground level when no sound is detected, and would generate a pulse reaching voltage

level of at least ±200mV when a beat is detected. However, the ±200mV pulse generated is not a square wave, and the voltage level is too lower for the input capture of our microcontroller to detect. Therefore, we need additional circuitry to establish the interface.

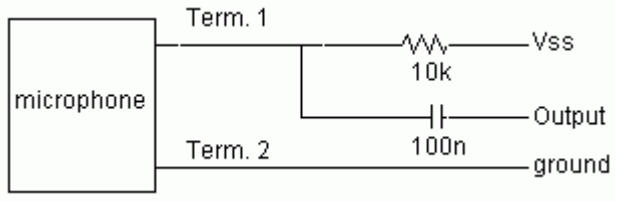

Figure 13: Signal Acquisition Circuit

4.4 Detection and Amplification

Detection of the pulse and the amplification of the detected signal are achieved by passing the acquired signal into a comparator and an inverting amplifier. The overall circuit diagram is shown in Figure 14.

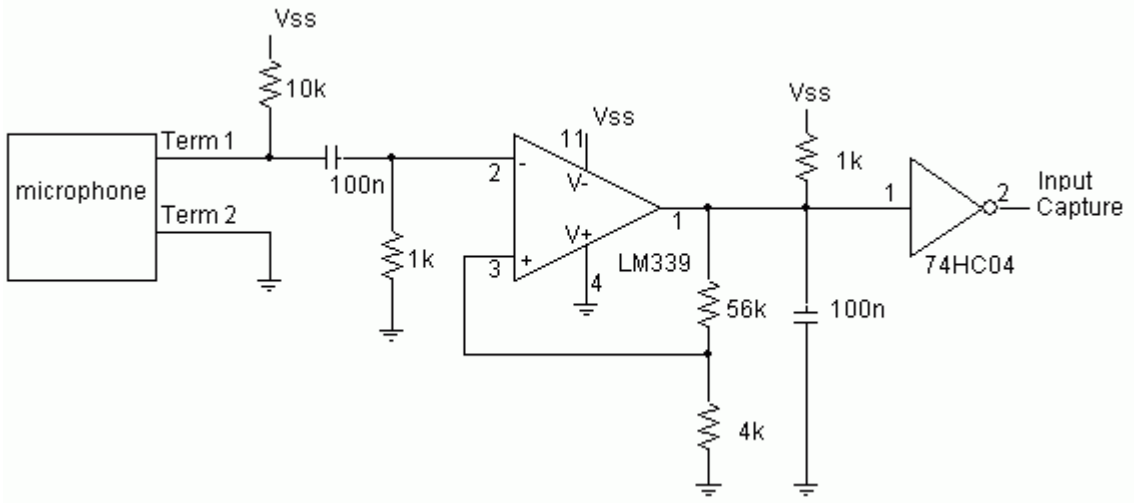

Figure 14: Pulse Sensor Overall Circuit Diagram

The hystersis included in the comparator allows us to eliminate jittering in the output waveform. The upper and lower voltage levels are set at ±200mV, and the open collector output of our comparator allows us to pull the voltage of our square pulse output to the desired level. With the comparator alone, output stays at its maximum voltage when no sound is detected and drops to zero when a heart beat is detected. However, the input capture is activated on the rising edge of the waveform, thus an inverted output is desired. Therefore, an inverter is introduced after the signal had been cleaned and digitized by the comparator to generate an output waveform that can be successfully detected by our microcontroller.

5. DESIGN OF TEMPERATURE SENSOR

This section details the acquisition of the user's body temperature, and the interfacing logic between the temperature sensor and the Rabbit 3000 microcontroller. As shown in the design overview, the temperature sensor consists of the MLX90601 IR Thermometer Module, and a control signal amplifier interface for communication with Rabbit 3000. This sector is shown again in Figure 15.

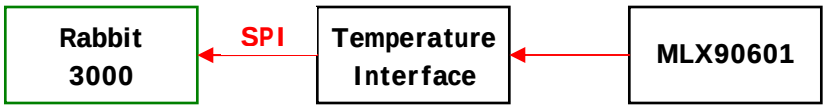

Figure 15: Temperature Measurement Block Diagram

5.1 MLX90601 IR Thermometer Module

In the interest of building a feasible prototype in our limited timeline, we choose to use an IR thermometer module to acquire temperature measurements. The specific module is MLX90601KZA-BKA with SPI interface available for communication. Since an analog to digital converter (ADC) is not included in the Rabbit 3000, the SPI interface is a convenient solution for successful communication. In addition, the flexible sensor design of the thermometer module, the sensor can be easily placed into the patient's ear for measurement. A detailed description of the module is provided in this section.

5.1.1 Architecture Overview

The internal circuitry of MLX90601 is shown in Figure 16, and the pin-out and pin descriptions are shown in Table 4.

As shown in Figure 16, the IR module consists of an IR sensor called MLX90247, which converts the heat flow into a proportional voltage output. The microcontroller MLX90313 then converts this voltage into a temperature reading through the embedded algorithm. The SPI ports, namely pins 7 through 10, allow Rabbit 3000 to obtain the temperature information from the module. The SPI protocol is discussed in detail in section 3.2.1.

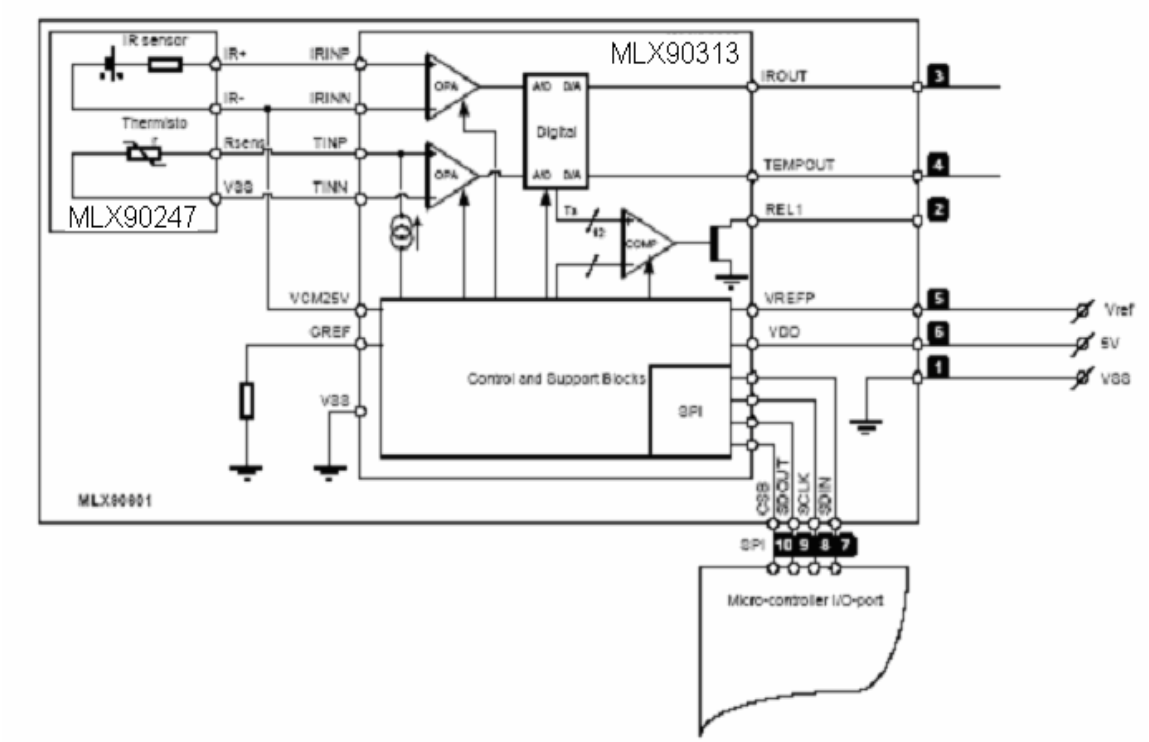

Figure 16: MLX90601 Internal Circuitry

Table 4: Pin-out and Pin Descriptions

Pin	Name	Function
1	VSS	Ground connection
2	REL1	Relay output
3	IROUT	Analog output infrared temperature
4	TEMPOUT	Analog output ambient temperature
5	VREF	Reference voltage output
6	VDD	Supply voltage
7	SDIN	SPI data in
8	SCLK	SPI clock
9	SDOUT	SPI data out
10	CSB	SPI chip select

5.1.2 Power Supply

The power is connected to pin 6 – supply voltage, and is drawn from the outlet from the wall with an AC adaptor to obtain a 5V power supply.

5.2 Control Signal Amplification

As described in previous section detailing our microcontroller, the SPI control output level of the microcontroller is at 3.3V. The voltage range is not high enough for the MLX90601 to interpret the control signals; therefore, we need to amplify the signal going into the SPI port of the temperature sensor module. The signal amplification circuit utilizing an open collector op-amp, LM339, is shown in Figure 17.

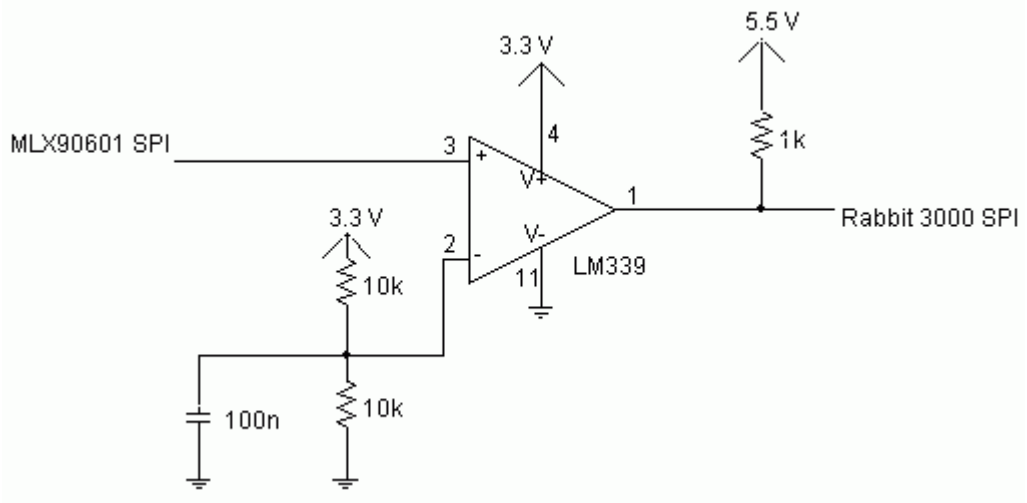

Figure 17: Signal Pull-Up Circuit Diagram

6. DESIGN OF DATABASE

Our database will be located in a server inside the hospital, and allows access from both client software (described in section 7) and the Rabbit 3000 microcontroller. The design the implementation of the database will be detailed in this section.

6.1 Choice of the Database

We choose to use MySQL 4.0 as our server database. The reason of choosing this database instead of writing our own is to save time and concentrate more on the hardware and client software components. Also, choosing an existing database helps the product to be more standardized as the format is compatible with many of the existing databases the hospital, and also the ease of manage, transfer, and merge data within the database.

6.2 Database Tables and Columns

In order to store all data needed for our product efficiently, we use four different tables. The first table is used to store the personal information of the doctor or patient, the second table stores the temperature data of all patients, the third table stores the pulse data for all patients; and the final table stores the patient-doctor association. The keys in these tables are described in detailed in Table 5-7 for the four tables [4].

Table 5: Keys of the User Detail Table

Key	Type	Attributes	Description
patient_id	unsigned int	primary, auto increment	Unique identifier for each user
type	bit	-	Determine user as doctor or patient[1]
password	char(128)	-	Password for the user, encrypted
first	char(128)	-	First name
middle	char(128)	-	Middle name
last	char(128)	-	Last name
machine	char(32)	(unique)	ID of currently associated machine
address	char(512)	-	Mailing address
hphone	char(32)	-	Home phone number
hfax	char(32)	-	Home fax number
ophone	char(32)	-	Office phone number
ofax	char(32)	-	Office fax number
mobile	char(32)	-	Mobile phone number
email	char(64)	-	E-mail address
emergency	char(512)	-	Emergency contact information
sin	char(32)	-	Social Insurance Number for patient
notes	char(10240)	-	Additional notes

[1] For this key, use 0 for patient and 1 for doctor

Table 6: Keys for Data Tables (Temperature and Pulse)

Key	Type	Attributes	Description
entry	unsigned int	primary, auto increment	Entry ID
patient_id	unsigned int	-	The unique identifier for the patient
time	timestamp	-	Time when the data is stored[2]
machine	int	-	The machine that is used for measurement
value	int	-	The value for the measurement

[2] Stored automatically when the data is written into the database

Table 7: Keys for Doctor-Patient Association Tables

Key	Type	Attributes	Description
entry	unsigned int	primary, auto increment	Entry ID
patient_id	unsigned int	-	The unique identifier for the patient
doctor_id	unsigned int	-	The unique identifier for the doctor

The design for the data tables allows a large amount of measurement data to be stored in the database without making it too complex. The design for doctor-patient association table allows many-to-many associations with ease in implementation.

7. DESIGN OF CLIENT SOFTWARE

Our client software, MediNet, written in C, allows both doctors and patients to access to database and view the measurement data. The design and implementation of the software is detailed in this section.

7.1 Basic User Interface Layouts

The user interface for the main windows will utilize the standard design used by most other software operating under Microsoft Windows, namely, it contains a menu bar, toolbar, workspace, and status bar. Menu bar contains access to all features, and the features are characterized into the categories: File, Edit, View, and Help. The toolbar gives quick access to the key features. The workspace displays all information for the windows, for example, a list of patients for doctor or a list of measurement data for the patient. The status bar displays the operating condition, including database connection status and data loading status. A sample user interface of the patient list for doctors is displayed in Figure 18.

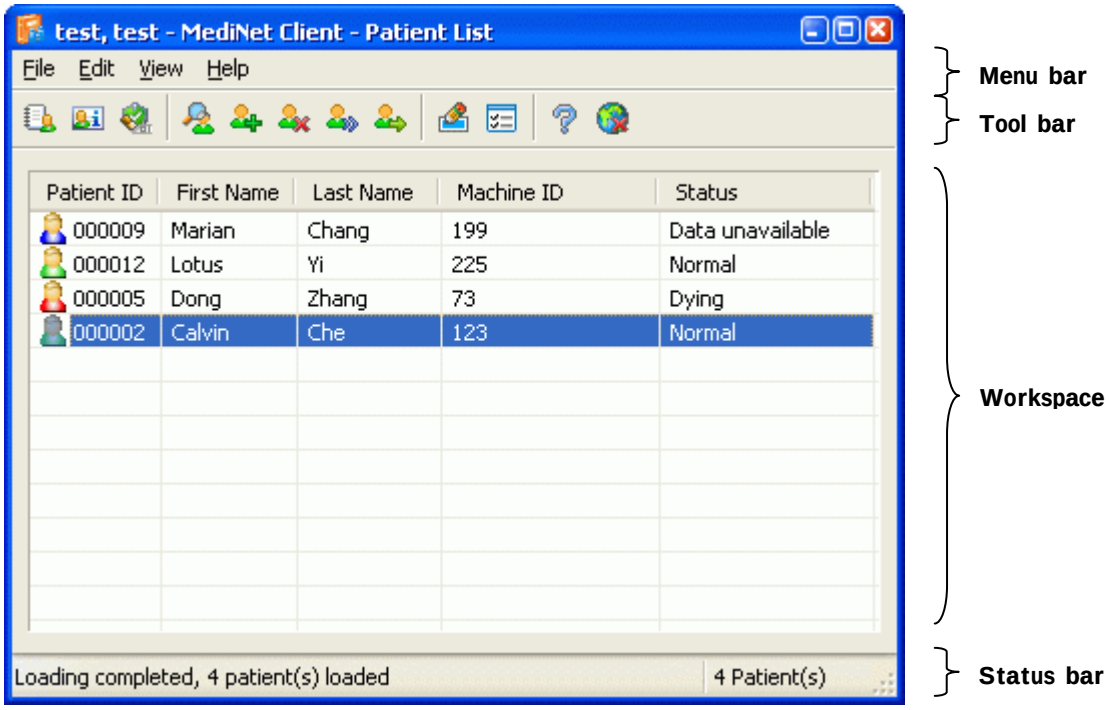

Figure 18: Main Interface for MediNet Software

7.2 Operation Modes

MediNet can operate under doctor or patient mode, and each mode has a different access privilege and operating functions. The determination of the mode is done by using the login ID to retrieve the user type from database (see section 6.2, Table 5).

In doctor mode, a list of patients will be displayed after login, and the doctor can view the data for all of his/her patients. Doctors can also edit their own personal information, as well as manage their list of patients. On the other hand, patients only have access to view their own measurement data and change their own personal information.

7.3 Database Access

All database access from MediNet will be done using the C API included in MySQL installation package. This API includes all functions that are necessary for database access, and no additional feature is required. All database access will be placed inside a thread so there will be no lock-up of the program due to network delay.

7.4 Description of Key Operating Features

7.4.1 Password Encryption (patient and doctor)

All passwords sent from the MediNet client will be encrypted by Md5 algorithm so the chance of exposure of personal privacy is decreased. This encryption is implemented by using pre-written codes.

7.4.2 Network and Program Options (patient and doctor)

The program allows user to change the address of the database server as well as many settings that govern the operation of the program. Both before login and inside the program, user is free to set the server address and connecting port, so that moving and changing of the server will produce minimum effect on the user. Other options include setting the time in between the automatic update of new data from server, as well as changing the default amount of historical measurement data to display. All settings are kept in a configuration file under program directory instead of using Windows registry to make the program clean and portable.

7.4.3 Patient Status (doctor mode only)

Once logged in, doctors can view a quick summary of measurement data of all their patients. This summary is displayed in the “Status” column on the patient list, as well as the patient icon. The status text is generated by comparing the last measurement data for both temperature and pulse with the typical value, and determining whether the data falls within the normal range. Texts such as “Body temperature too high” will be

generated if an outbound data is observed. The color of the patient icon change according to this summary information, with green indicating normal, yellow indicating warning, red indicating critical condition, and blue indicating data unavailable.

7.4.4 Patient List Management (doctor mode only)

Doctors can modify the doctor-patient association table (section 6.2, table 7) by adding, removing, and transferring patients to or from their list. Adding patient creates new entry in the table, which involves searching the database by specifying patient ID or name, and all matching result will be displayed in a separate dialog. The similar method applies to transferring the patient with the targeting doctor ID or name, and this involves modifying the association table entry. Removing patient from the list is done by simply deleting the corresponding entry in the association table.

7.4.5 Edit of User Information (patient and doctor)

Both doctor and patient are free to change their personal information. The "Edit Information" dialog will first load all information from the server database in order to keep the information current, and will replace all information fields with the new data regardless if it is changed or not. All modification of personal information requires user password to access to prevent others from modifying the data.

7.4.6 Changing of Machine Association (patient and doctor)

The association between patient and machine can be changed in order to share a machine between two or more patients. Each microcontroller is pre-loaded with a machine ID with their firmware, and will use this ID to access the database. As described in section 3.4.3, in order to successfully write the new data into the database, the server software must find a matching ID to retrieve the patient ID for updating the value.

To change this association, a direct connection between the client software and microcontroller board is required, and is implemented using the double handshake algorithm, and is illustrated with Figure 19. This connection is made to obtain machine ID, and also serves as a way to prevent user associate him / herself to the wrong machine leading to false writing of data. Furthermore, the machine and patient is a 1-to-1 association, which means that once a new association is created, the old one will be removed.

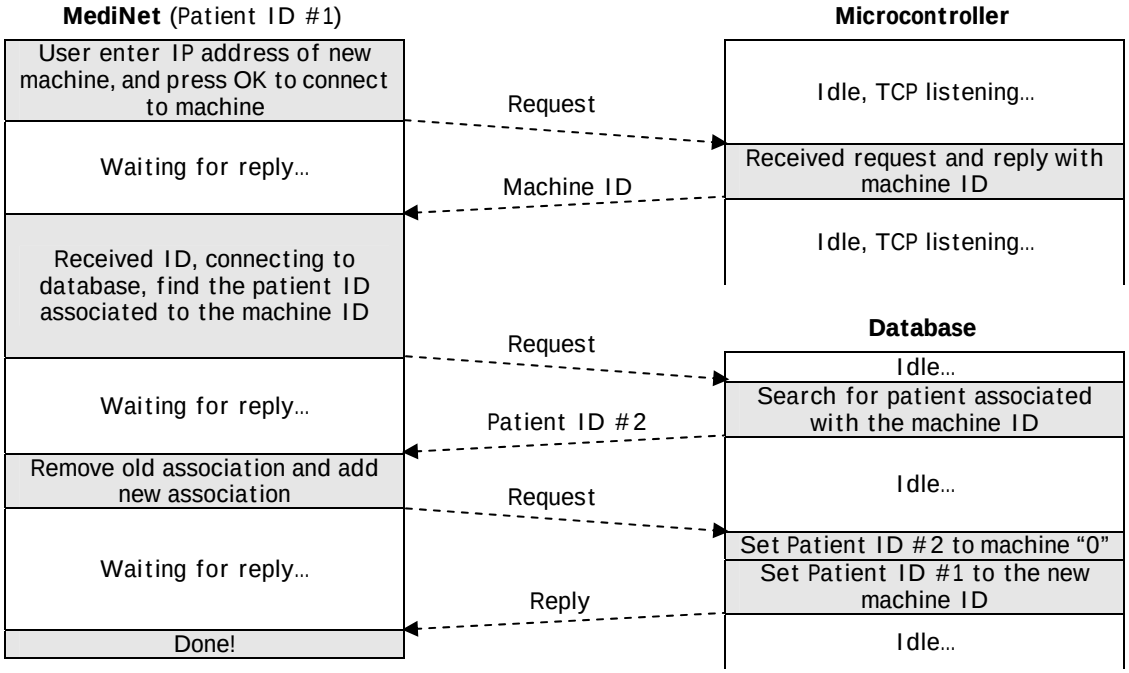

Figure 19: Mechanism for Changing Machine Asociation

7.4.7 Viewing Measurement Data (doctor and patient)

The measurement data for each patient can be retrieved from the server and displayed in two lists: temperature and pulse. The lists will contain three columns: time of measurement, the ID of the machine used for measurement, and the actual measurement value. The amount of data to be retrieved can be specified in the option, and can be set to retrieve a fixed amount of most recent data, or all the data that are taken within the past few days. Both doctor and patient have no permission to modify any of these data in order to retain its accuracy.

8. CONCLUSION

The main objective for our design is to maximize efficiency as well as flexibility. From the user's perspective, an efficient health measuring device will ensure the accuracy and responsiveness to any measurement, which is a crucial aspect to any medical instruments. As designers, we are striving to conceive a device that is as flexible as possible in terms of desired future enhancement, which will be built on top of the current prototype. Either from the hardware or software point of view, our design approach ascertains that the product will remain feature enriched, thus competitive in the market. We are confident that, by employing the design specifications outlined in this document we can achieve a balance between user satisfaction and the effort required in implementation, in other words, these design procedures maximize the performance cost ratio.

9. SOURCES AND REFERENCES

[1] Rabbit Semiconductor, “Rabbit 2000 TCP/IP Development Kit,” 2005, http://www.rabbitsemiconductor.com/products/rab2000_tcpip/index.shtml.

[2] Melexis, “MLX90601 family – IR thermometer modules”, 2002, http://www.melexis.com/prodfiles/mlx90601.pdf.

[3] EMKAY, “MD9745APA-1 Product Specification”, http://rocky.digikey.com/WebLib/Emkay%20Products/Web%20Data/MD9745APA-1.PDF.

[4] MySQL Reference Manual, 2005, http://dev.mysql.com/doc/mysql/en/index.html.