

Abstract

In this project the goal was to design a patient monitoring system that will monitor the patients critical information(temperature and heart rate) and upload them into the data base, and in case of emergency or patients request, it will send a portion of the record to doctors computer.

Introduction

As many other applications, real-time embedded systems can be used in relation to dealing with patients and their medical issues. Patients that are in the hospital are periodically checked by nurses for their situation and some critical information related to their health, and if there are any critical data observed, doctor will be informed about the problem. Also sometimes patient might need to request doctors help.

We designed a system that is much easier and faster and can eliminate the need for having someone to check up the patient periodically.

Considering this I implemented a system that would collect periodically generated health data from devices like heart rate monitor and upload them into the FairCom database. In case of receiving critical data or patients request (by pushing a button) last 10 rows of data (retrieved from database) will be sent to doctors machine as a message. The main components of this system are the user space program for patients machine and kernel space program that detects the button push for patients request, and also doctor machines user space program that will receive the messages of patients and collect them into a file.

Background

For implementation of this system I used mostly any concepts that was covered during the labs.

Having data that are periodically generated by measurement devices, I used real-time tasks to first simulate these measurement devices, and then keep track of these data's to upload them into the database using another real-time task.

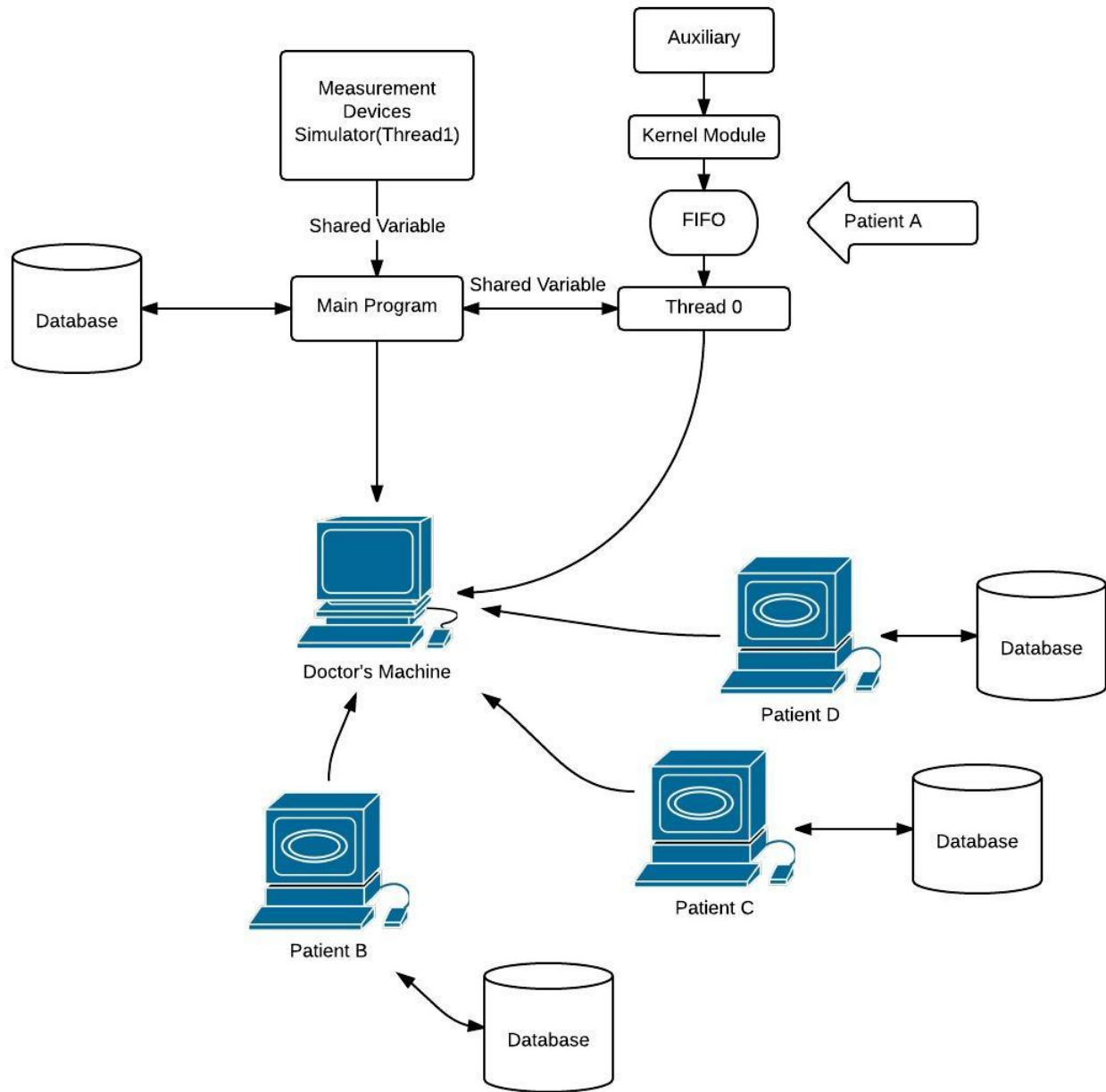
There are two threads and the main program communicating with each other in the patient system's program and also the user-space and kernel space programs are communicating with each other, And lastly doctors machine is a remote machine that my patient machines communicate with via passing messages. So almost many communication means are used for inter task/thread communication, and this arose some concerns about mutual exclusion that is dealt with using mutex.

Also hardware interrupts are used to detect the push button right away on port B as what we had in lab6.

And lastly FairCom data-base that is an embedded system data base commonly used by embedded system designer is used to collect data.

Proposed method/System description/Implementation

As already mentioned the design consists of 5 major elements that are the patients user space and kernel space program and also the doctor machines program.



Shown above is the over all overview of the system. As shown each patient machine has its own data-base server and they communicate with the remote doctors system.

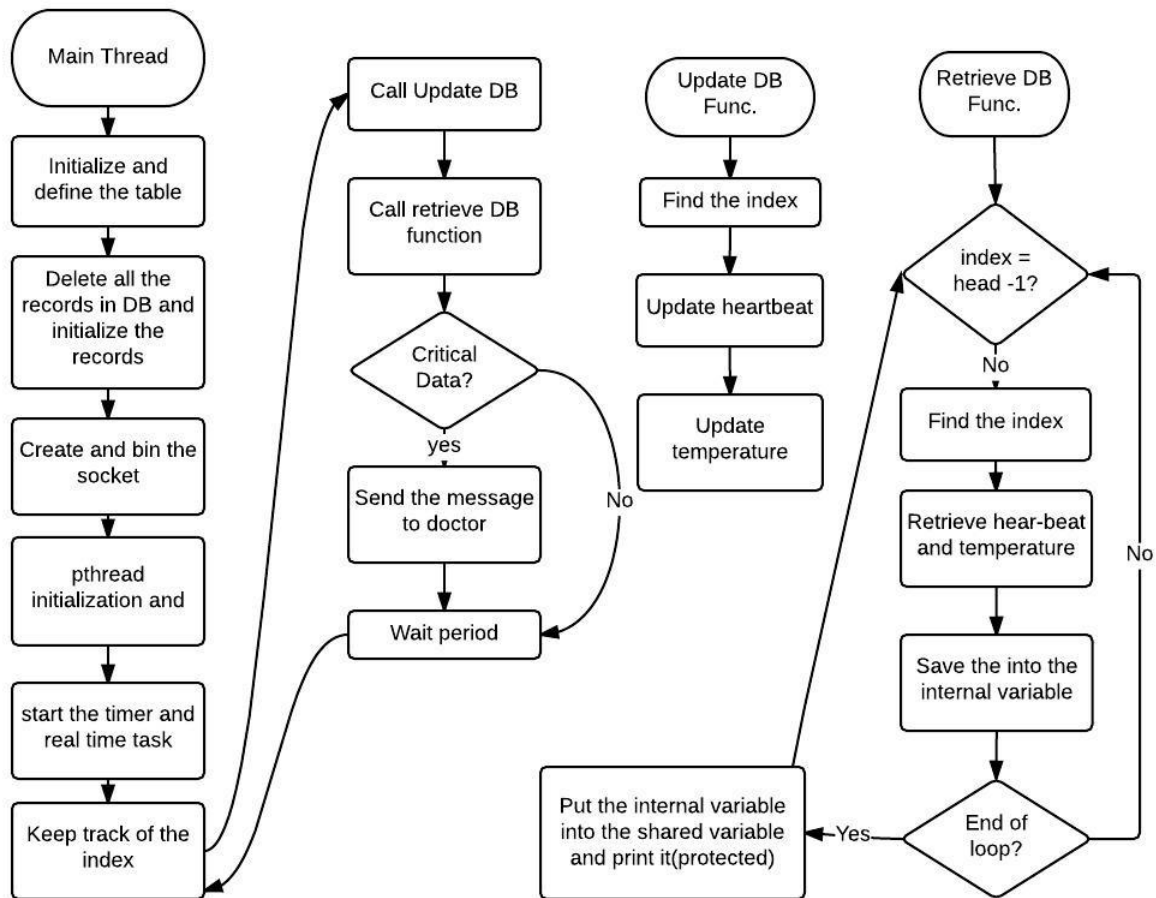
The diagram located in the upper part of the picture represents the internal interconnections between tasks in Patient A's system. The main program is the main task that interconnects with database to upload data into or retrieve data from and also checks for the critical data as a real-time task. Thread1 is responsible for simulating measurement devices like heart rate and temperature monitor and it shares the generated

data with other tasks with updating them into the global variables.

Thread 0 is responsible for listening to FIFO for push buttons that are sent from kernel module. And if detects any it would send patients help request along with the last condition data to the doctor.

The kernel module is responsible for tracking the push buttons from the auxiliary board, connected to TS7250 board using GPIO interrupt registers for port B and signaling the user space program via FIFO.

The flow charts below specifically represents the function of each tasks.



The main thread in patient's user space program first initiates and defines the Data base tables using Define() and Initialize() functions that are provided by the FaiCom tutorial, then the functions related to deleting every record in the data base and after that initializing records to it, would be called. The Table that I create consists of three fields, namely index, heartbeat and temperature, and I filled them with zeros except for indexes that I search the table based on them, I started the indexes from 100 and my table is capable to store 30 recorded generally in the table at the moment.

So these are the Function used from far come:

Initialize() : Perform the requirements of logging in to the ctree server

Define(): Opens the table if exist other wise it creates and opens it(I modified this, but I just changed the name of the table to "mytable1" and also I added my fields (in the

Create_CustomerMaster_Table)).

Delete_Records(): Will delete all the records in the table (I didn't modify anything)

Add_CustomerMaster_Records(): Adds records to table from an array of strings that I set to be all zeros except for indexes.

Then the exact approach as lab 5 and 6 will be pursued for getting the port numbers and Ips And initializing them to the relevant data structures to have information about host(patients system) and the computer that we want to send data to (Doctor computer). Then after initializing the socket and binding the socket to data structure and getting the host info using gethostbyname function and so on, threads and the mutexes are initialized and created and after all these initializations in the real-time portion of the program each time the index would be increased and the update function would be called. This is a modified version of FairCom update function in which we would search the index field for specific index and update next two filed records there.

After that we would also retrieve last 10 records from table using myget_CustomerMaster_record() function that is a modified version of Faircom update function but I protected this function and also the process of adhering critical info. Statement to it to (in case of having critical info.), make sure the message would not be corrupted as it is a shared variable. If the data is critical the message also would be sent to doctor in the protected area.

And then the real time task would wait until its period.

I use 10000000 ns as base period and $80 * \text{baseperiod}$ for the period of my both realtime tasks.

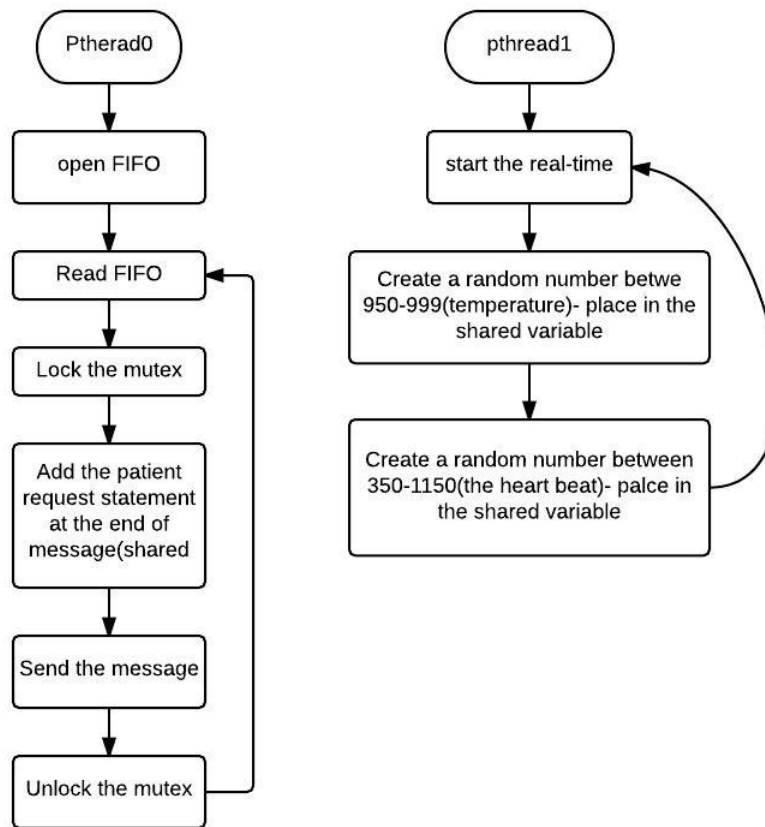
Also its important to mention that the IP of the doctors machine is hardcoded in the program.

In the picture we can observe the flow chart for myUpdate_CustomerMaster_Record() that I discussed before.

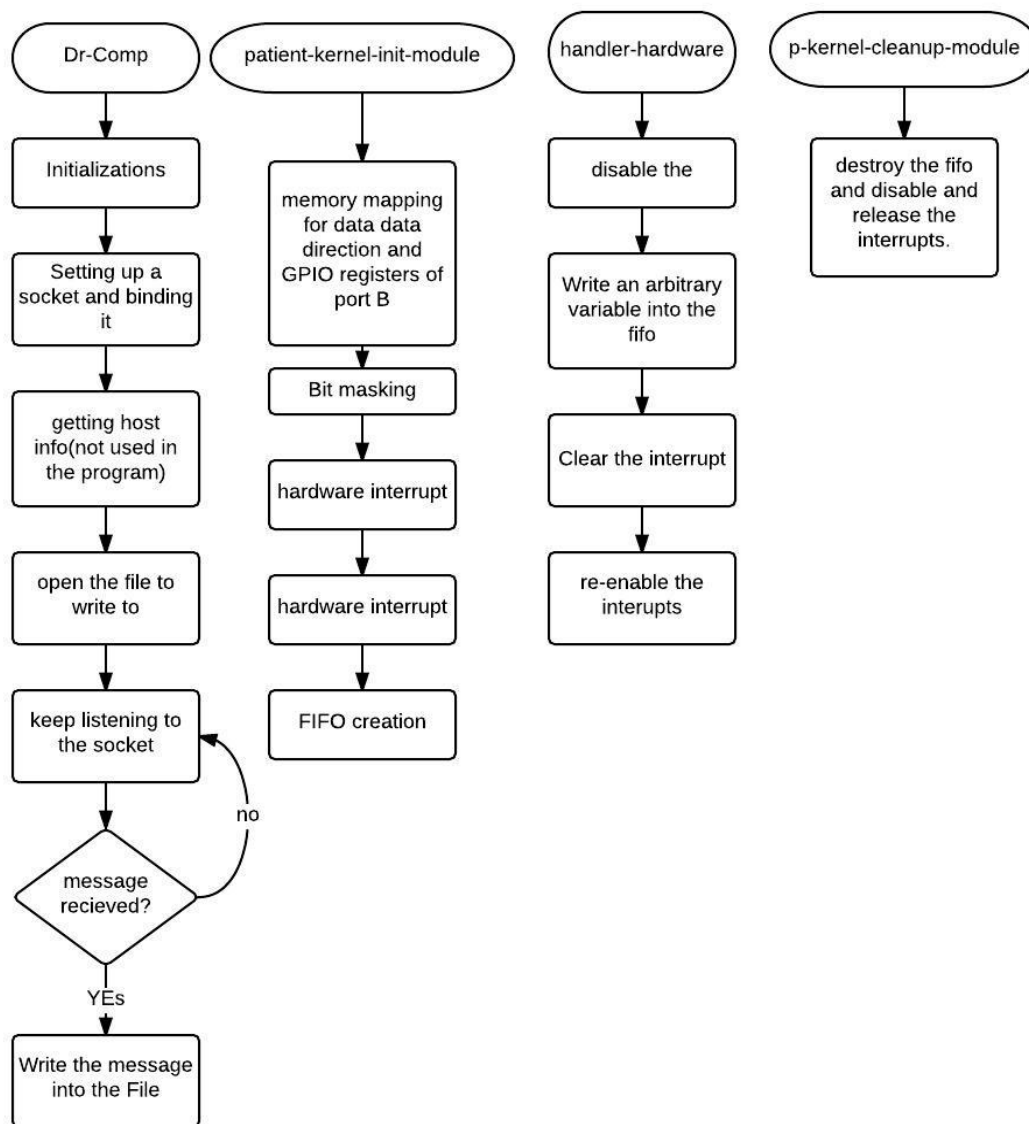
In the myget_CustomerMaster_Record() function I search again based on the current index and then fill a variable with the content of two next fields for the last 10 records. And then I copy the content of the variable into the message variable that is our shared variable in a protected area by mutex.

Below You can see the flow charts related to pthread 1 and 0 of the patients user space program. As mentioned before Pthread 0 opens and reads FIFO and in case of receiving data from kernel it would stick the patient request statement to the shared message variable and send it to doctors machine in a protected area.

And pthread 1 is a real time task is a realtime tasks resembling measurment devices that creates data periodically. The random data generated for heartbeat is between (999-950), While for the temperature is between(1150-350), and the normal range that is checked in the main thread is considred to be between (1100-400) and (990-960) respectively.



Below is the flow charts related to the kernel module for the user space program and also doctors computer program. The purpose of doctors computer program is to get the messages from patients, by using sockets and the exact way we did in lab 5 and 6. And the purpose for kernel module in patients system is to detect hardware interrupts using GPIO registers of portB and in the handler function after disabling interrupt an integer variable would be written into the fifo and the interrupt would be cleared and interrupt would be enabled again. The whole procedure is shown below.



Experiment and results

For experimenting the correct function of the program in each step of developing the design I tested that in different ways like printing the shared variable in different parts to see the changes in it or displaying the content of the table to track the changes and so on. After completing the program to make sure that my patient user space program is working fine I had the printing statement for the message shared variable in the myget function and also before sending it to the doctor as a result of patient request or critical condition. And meanwhile I displayed the content of table to make sure the true data is being sent. First I did not have any critical condition statement and all I observed was the push button statement with the message, and that was because of the wrong checking statement. Afterwards, running the doctors program on another board I tested my program completely in a way that whenever it detected the critical data, doctors machine received past 10 records of data with the critical info. Statement and when I pushed the button the message had the past data and patient's request statement.

The message recieved is

temperature = 958	heartbeat= 379
temperature = 997	heartbeat= 587
temperature = 986	heartbeat= 387
temperature = 950	heartbeat= 385
temperature = 984	heartbeat= 527
temperature = 991	heartbeat= 404
temperature = 968	heartbeat= 797
temperature = 986	heartbeat= 708
temperature = 985	heartbeat= 788
temperature = 989	heartbeat= 663

*****a critical info received from(10.3.52.18)*****

The message recieved is

temperature = 981	heartbeat= 575
temperature = 958	heartbeat= 379
temperature = 997	heartbeat= 587
temperature = 986	heartbeat= 387
temperature = 950	heartbeat= 385
temperature = 984	heartbeat= 527
temperature = 991	heartbeat= 404
temperature = 968	heartbeat= 797
temperature = 986	heartbeat= 708
temperature = 985	heartbeat= 788

*****The patient on (10.3.52.18) needs help*****

The message recieved is

*****The patient on (10.3.52.18) needs help*****

The message recieved is

temperature = 968	heartbeat= 942
temperature = 981	heartbeat= 575
temperature = 958	heartbeat= 379
temperature = 997	heartbeat= 587
temperature = 986	heartbeat= 387
temperature = 950	heartbeat= 385
temperature = 984	heartbeat= 527
temperature = 991	heartbeat= 404
temperature = 968	heartbeat= 797
temperature = 986	heartbeat= 708

*****The patient on (10.3.52.18) needs help*****

The message recieved is

*****The patient on (10.3.52.18) needs help*****

The message recieved is

*****The patient on (10.3.52.18) needs help*****

The message recieved is

*****The patient on (10.3.52.18) needs help*****

The message recieved is

*****The patient on (10.3.52.18) needs help*****

In the picture above at the first message we have 958 as the temperature that is less than 960 so critical. Then the push button is detected and the last row of data is not critical, then the patients pushed the button again within the same time period in realtime task and as the data was sent before just the patient request statement would be printed. And if the patients keeps pushing the button this statement will be sent over and over again until we have new data in the table so the data is also sent with patient request statement. It May happen that the new data is critical so the data with critical message is sent and then the patient also request help, in this situation a message with both data critical info and patient request statement would be sent to the doctor as shown below.

```
The message recieved is
  temperature = 994      heartbeat= 552
  temperature = 998      heartbeat= 587
  temperature = 956      heartbeat= 1028
  temperature = 970      heartbeat= 433
  temperature = 957      heartbeat= 977
  temperature = 991      heartbeat= 870
  temperature = 973      heartbeat= 580
  temperature = 992      heartbeat= 1028
  temperature = 959      heartbeat= 1008
  temperature = 995      heartbeat= 909
*****a critical info received from(10.3.52.18)*****

The message recieved is
  temperature = 994      heartbeat= 552
  temperature = 998      heartbeat= 587
  temperature = 956      heartbeat= 1028
  temperature = 970      heartbeat= 433
  temperature = 957      heartbeat= 977
  temperature = 991      heartbeat= 870
  temperature = 973      heartbeat= 580
  temperature = 992      heartbeat= 1028
  temperature = 959      heartbeat= 1008
  temperature = 995      heartbeat= 909
*****a critical info received from(10.3.52.18)*****
*****The patient on (10.3.52.18) needs help*****
```

But in case the patient had pushed the button again we would only sent the patient request statement as the data in shared variable “message” would be cleared within each push button.

I ran my program many times and tried to push the button in different times and keep pushing it sometimes.

So there happened sometimes that I kept pushing the button and within the patient request statement one critical info. Statement was printed, the reason was related to the fact that after calling myget function that puts the data in message(shared var.) I checked the critical condition and sticked the statement to message and send it. But If a push

button happens at the time that update is called and the statement is not adhered to that yet, thread0 will get the message because of push button and change it. So I used mutex and put the whole calling of myget and sticking critical statement, and sending process into the critical section. But this time my program was blocked, and that was because in myget function I again use mutex to protect writing into message so I created another mutex(mutex1) to not to get blocked in myget for ever. Below is a snap shot of the patient user space program window, however some of printed statements was redundant and just for debugging purposes that in a version that I submitted I eliminated them.

I also tested the program with having more than one patient at the moment on the port (two patients) and the result(doctors window and two patients window) is presented in the video that I attached to this report.

Two windows on the left are patients windows(on board number 15 and 18) and the one at the left is doctors window. Also the log file that is saved in doctors machine is attached to this report that is "file.txt".


```
heart beat , tem, index in myupdate are 649,979 116
    Update record...
        with out Press <ENTER> key to unlock, unlocked and finished update
starting retrieve function with index= 116
end of retrieve
message id
temperature = 979      heartbeat= 649
temperature = 966      heartbeat= 1064
temperature = 963      heartbeat= 964
temperature = 971      heartbeat= 475
temperature = 972      heartbeat= 1069
temperature = 965      heartbeat= 475
temperature = 997      heartbeat= 921
temperature = 991      heartbeat= 836
temperature = 961      heartbeat= 571
temperature = 998      heartbeat= 397
```

```
heart beat , tem, index in myupdate are 797,957 117
    Update record...
        with out Press <ENTER> key to unlock, unlocked and finished update
critical data
starting retrieve function with index= 117
end of retrieve
message id
temperature = 957      heartbeat= 797
temperature = 979      heartbeat= 649
temperature = 966      heartbeat= 1064
temperature = 963      heartbeat= 964
temperature = 971      heartbeat= 475
temperature = 972      heartbeat= 1069
temperature = 965      heartbeat= 475
temperature = 997      heartbeat= 921
temperature = 991      heartbeat= 836
temperature = 961      heartbeat= 571
```

```
heart beat , tem, index in myupdate are 615,982 118
    Update record...
        with out Press <ENTER> key to unlock, unlocked and finished update
starting retrieve function with index= 118
end of retrieve
message id
temperature = 982      heartbeat= 615
temperature = 957      heartbeat= 797
temperature = 979      heartbeat= 649
temperature = 966      heartbeat= 1064
temperature = 963      heartbeat= 964
temperature = 971      heartbeat= 475
temperature = 972      heartbeat= 1069
temperature = 965      heartbeat= 475
temperature = 997      heartbeat= 921
temperature = 991      heartbeat= 836
```

```
heart beat , tem, index in myupdate are 597,981 119
    Update record...
```

Discussion and Conclusions

I discussed some of the cases that I tested above but additionally, As many boards were mostly Occupied by others I couldn't test my program with more than two patient at the moment, and also as we do not have many boards I couldn't test it with many patients. But in case there were a huge traffic on the port that my doctor is using to and the network congestion happened, (like having hundreds of different programs sending messages through that port to doctor), I think as we defined the protocol to be UDP we could have lost some of the messages sent by the patients.

Also in my implementation each patient has its own data base so in case of any damage or problem the whole data would be gone, but we would have the critical data saved on the log file in doctor's computer. But it could be better to have a data base on doctor's computer and send all data in there too, so that doctor would be able to access all data in any amount.

Also as the medical information is related to the age and other factors, the range that I chose for generating data and the range that I considered as normal range would differ based on the application and also the time period for real-time application is also related to the application. And Also I saved just 30 rows at the time in the table and retrieved 10 of them that this also could differ based on the real world , and medical requirements.

In relation to FairCom data base I encountered problems like with indexing, fair come used some string like variables to search table for and when I wanted to converted my integer indexes to strings the different length of 1 indexes with 2 digit variables was problematic, So I started from 100 as 0 to solve my problem. I also had problem with updating or retrieving the variables for more than one thread at the same time(That later I found out that I did not necessarily needed to do so) and I was suspecting that the problem is in relation to the internal block function that is used for those functions, So I commented the block and unblock function but the error was propagated to the next function. I solved the problem by havving message shared variable that always holds the last 10 elements in the table so in case of push button it can be sent to the doctor right away.

Also while running the server as more than one patient w would have error if the name of table is the same, so for each of the patients the name of the table should be changed.