Asian Journal of Information Technology 6 (10): 1050-1056, 2007
ISSN: 1682-3915
© Medwell Journals, 2007

Optimization of Reliability in a Real Time Security System Controlled by Embedded Internet Technology

[1]Shah Mostafa Khaled, [2]Rezaul Karim, [2]Ashis Kumer Biswas, [2]R.H. Rahman, [2]Nusrat Nowsheen and [3]Mohammad Abdul Qayum
[1]RCITE, Legal and Judicial Capacity Building Project, Ministry for Law, Justice and Parliamentary Affairs, Bangladesh
[2]Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh
[3]Rollout Field Engineering, Network Deployment, Banglalink (An Orascom Tel. Comp.), Dhaka, Bangladesh

Abstract: This study addresses the reliability optimization issues of a real time security system developed by Embedded Internet technology. This study describes three solution approaches to address tradeoff between speed of request processing and reliability of processing requests. Main intention here is to make the system work at real time and ensure reliability of the system at the same time. Among the three approaches significant results were observed for Multiprocess Approach Using Time Delay, which works especially better number of message in the SMS Inbox is higher.

Key words: Embedded internet control, reliability, real time system, SMS, multiprocess

INTRODUCTION

System reliability, defined as the probability of a system functioning correctly, refers to the degree of tolerance against errors and failures in a system (Painton and Campbell, 1995). It may also be defined as the ability of a system to perform its required functions under the stated conditions for specified period of time. The reliability is defined in statistical terms as "the probability of failure-free operation of a computer program in a specified environment for a specified time" (Musa *et al*., 1987). The reliability of a system may be calculated from failure statistics for a number of samples of the unit.

A real time software system is a system where the correct functioning of the system depends on results produced by the system and time at which these results are produced (Sommerville, 2003). One way of looking at a real time system is as a stimulus/response system. Given a particular input stimulus, the system must produce some corresponding response within a given acceptable time (Burns and Williams, 1997). In other words, a real time system has less time wastage in processing of information. It performs its assigned task within least possible time within tolerance.

In developing a general hardware, OS and software tools for distributed, Internet-enabled, PC-based embedded control systems, Feng *et al.* (2002) faced that a modern embedded control system requires real-time machine vision, sensing and motion control. It requires extensive development, such as the advanced highway maintenance to achieve a good degree of reliability. The design of a perfect computer control system requires that it should respond to external signals within a finite and specified period of time, i.e., it should be real-time and for such system it is important that the results are correct to attain a full-scaled reliability (Bajorn). The research of Lloyd and Susnik (2002) suggests that using a field instrument or sensor in developing embedded web server simplifies maintenance, diagnostics and in-place firmware upgrades. This improves the reliability of the embedded system. Another important work in developing a good real time reliable system described in David *et al.* (1997) where dynamic reconfiguration of an embedded control system improves its real-time features and increases the reliability. NASREM (1989) is an example of such architecture. To develop complex real-time systems, it is necessary to use software tools to develop and analyze embedded real-time applications. Software frameworks can be successfully used for such applications (Cechticky *et al*., 2002; Cechticky and Psetti, 2003; Pasetti, 2002). Some real-time frameworks have been developed, such as the work reported in (Hassan *et al*., 2002) where a framework to develop real-time robotic applications is introduced. In Leslie *et al.* (1998) a framework to reuse components in real-time multi-processing systems is presented.

Corresponding Author: S.M. Khaled, RCITE, Legal and Judicial Capacity Building Project, Ministry for Law, Justice and Parliamentary Affairs, Bangladesh

This study addresses the reliability optimization issues of a real time security system developed by Embedded Internet technology. In this study, reliability of the system is measured in terms of error free and expected behavior of the system. The system works reliably if it does not get hanged and if it successfully processes the control request messages. The real time property of the system is measured in terms of how fast and error freely the system can respond to control request messages. In that meaning, reliability is also encompassed in the real time property. However, the project under this study has a tread off between working at the best possible speed and processing control requests reliably.

This study addresses Real Time issues of a security system controlled by Embedded Internet technology, the full project is presented in Khaled and Karim (2006). In this system, a sensor and an actuator, interfaced with a client PC connected to the Internet, is set up in the door of a room as shown in Fig. 1. This PC communicates with a security server through the Internet and a mobile telephone is interfaced with the server, as shown in Fig. 2. When the door is opened the sensor will generate a signal to the client PC and the PC will send a door opening information request to the server. Server will send an SMS message, through the gateway mobile interfaced with it, to the mobile phone of the room-owner about the opening of the door. If the owner of the room wants to close the door, he has to send a closing request SMS to the gateway mobile interfaced with the server. Upon receipt of this SMS the server issues a door close request to the client. Receiving this request the actuator is activated and the door is thereby closed. This study is only concerned about the retrieval of SMS messages from the SMS Inbox of the gateway mobile interfaced with the server.

The purpose of mobile phone interfacing in this project (Khaled and Karim, 2006) is to generate SMS messages directly from server and to receive SMS messages in the server. A mobile telephone device (Nokia 3310) is interfaced with the server machine that is used to send SMS message to the owners of the rooms about door opening and to receive closing request. Thus the mobile interfacing does two particular things: Sending SMS, Receiving SMS. What the mobile interfacing program actually does is that it reads the SMS Inbox of the mobile set, to see whether there is any new door closing request. If there is a new closing request SMS, then the request is saved in a database. Nokia 3310 provides with a binary command set, F-Bus Command Set, (GSM, Forum Nokia) which enables us to send and receive messages by sending binary frames to Serial (COM) port. Figure 3 describes the operation of the mobile accessing program. The kits necessary for such interfacing are Nokia 3310 mobile set and F-Bus Connector. Figure 4 shows these components along with the F-Bus pins of Nokia 3310 set.

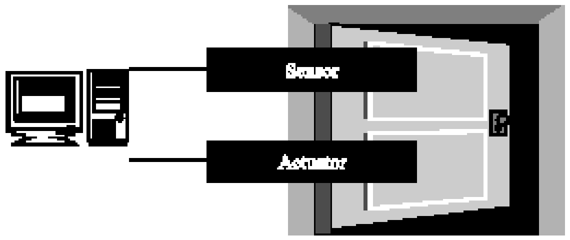

Fig. 1: Inside a room: Sensor, actuator connection with client

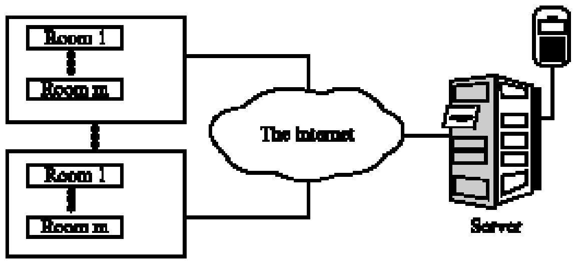

Fig. 2: Server client communication

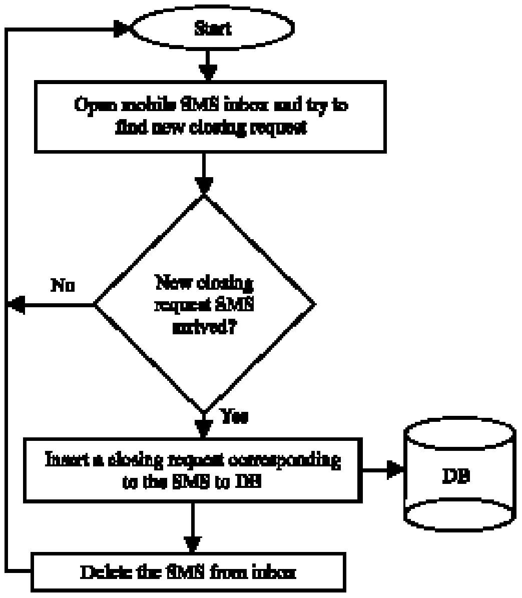
Start
Open mobile SMS inbox and try to find new closing request
New closing request SMS arrived?
No
Yes
Insert a closing request corresponding to the SMS to DB
DB
Delete the SMS from inbox

Fig. 3: Server-mobile communication

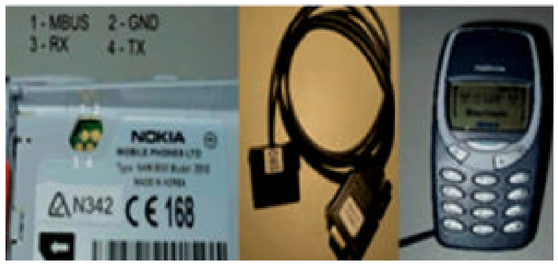

Fig. 4: F-bus pins, f-bus connector, Nokia 3310 mobile

Sending and receiving SMS (Khan *et al.*, 2007) from the server computer through the Nokia 3310 mobile set has some problems. The hardware equipment Nokia 3310 mobile set is accessed through a data cable called F-Bus Connector (Khan *et al.*, 2007) which is a serial interface and is inherently slow, on one hand. On the other, the SMS Inbox of the mobile phone is a non-sharable resource. If it is tried to access using multiple processes, there is every likelihood that one process may try to read a particular SMS while another is trying to delete it. This may cause the program malfunction. What actually happens inside Nokia 3310 SMS Inbox is not known. However, from a user's point of view the authors of this paper observed that the program gets hanged in such cases. This resource is not perfectly suitable for a multiprocess solution. As a result, there is a tradeoff between reliably retrieving the SMS messages from the mobile and making the system work at real time. The utmost necessity is to ensure that all the closing request SMSs messages are reliably retrieved and the system works considerably fast at the same time.

This study describes three solution approaches to address this tradeoff. The prime intention is to keep the system as fast as possible, i.e. to reduce any sort of delay and thereby make the system work at real time in the best possible way. The reliability of the system, at the same time, must be ensured so that the system does not fail to process any door closing request in the quest for better speed.

As methodology of analysis, the system was physically tested, rather than doing any simulation. Using the collected data, a statistical analysis has been made to compare different algorithm's performance. The analysis produced an optimized delay between attempts to retrieve messages.

APPROACHES TO MAKE THE SYSTEM A REAL TIME

Three approaches to read SMS from mobile SMS Inbox has been described. In the first approach a loop has been used to sequentially read SMS messages and write those to a database. The second approach is a multiprocess solution and a semaphore is used to synchronize use of mobile SMS Inbox among different processes. The final approach is also multiprocess solution, where a time delay is used instead of semaphore.

Sequential approach: In this approach, retrieving SMS and further processing is done sequentially. The program checks the SMS Inbox entries one after another. The pseudocode of sequential approach is given in Algorithm 1. Inside a single loop, the SMS Inbox is opened and a close request SMS is searched. If such a SMS is found, it is read, inserted into the database, then it is deleted; the SMS Inbox is then closed. The loop iterates this sequence of activities and thereby processes closing request SMS.

```
While (true) {
     Open SMS Inbox();
     For (each close request message in SMS Inbox) {
         Read the SMS Message();
         Insert the message data into database();
         Delete this SMS from the SMS Inbox();
         }
         Close SMS Inbox();
}
```

Algorithm 1: Pseudocode of sequential approach

Although this approach is quite easy to implement and ensures full reliability, it is very slow. Multithreaded solution can boost the speed of message retrieval.

Multiprecess approach using semaphore: Multiprocess solution is used to reduce the average single SMS processing time. But it has been observed that if more than one process tries to access the SMS Inbox at the same time, then the system malfunctions. Therefore, to make the SMS Inbox a mutually exclusive resource for the processes, a semaphore has been used. Any process can access SMS Inbox only when it has locked the semaphore. Since only one process can access the semaphore at any instant, only one process can access the SMS Inbox.

The pseudocode of this approach is given in Algorithm 2. The processes sees whether the semaphore used is locked or free. If the semaphore is being used by another process, then the process has to wait until the semaphore is free. As soon as any process gets the control of the semaphore, it is exclusively eligible to access the SMS Inbox. The proces opens the SMS Inbox, tries to find any closing request, if there is a closing request SMS, the process reads it, saves it in a buffer, deletes the SMS message, closes the SMS Inbox and releases semaphore so that other processes may use it.

```
For all process $P_i$
While (true) {
     If(any process $P_j$ [i#j]is not using SMS Inbox){
      Lock (SemaphoreInbox);
         Open SMS Inbox();
         If(any close request in the SMS Inbox){
             Read the SMS message();
             Delete this SMS from the SMS Inbox();
```

```
            Close the SMS Inbox();
        Unlock (SemaphoreInbox);
        Insert message data into database();
            }
        }
}
```

Algorithm 2: Pseudocode of multiprecess approach using semaphore

This approach also ensures full reliability in the meaning that there is no chance of message lost, but the average time necessary for single message retrieval improves.

Multiprocess approach using time delay: Another multiprocess solution to use a delay between 2 process creation instead of using semaphore; so that one process tries to access the SMS Inbox. Here a process is created, than it access the SMS Inbox, reads the SMS messages to find any closing request, if found than reads it and deletes the particular SMS. The processing time for the all the SMS messages in the SMS Inbox can also be improved in this approach.

```
While(TRUE){
    after T second interval
    Create a new process(){
       Open the SMS Inbox();
       If(any close request in the SMS Inbox){
          Read the SMS message();
          Insert message data into database();
          Delete this SMS from the SMS Inbox();
          }
       Close the inbox of the mobile device();
       }
    Destroy the process();
}
```

Algorithm 3: Pseudocode of multiprecess approach using time delay

The pseudo code of Algorithm 3 presents an algorithm for using this mechanism. Every process is created after an interval of T seconds. Here selection of this interval T is very critical. If the interval is too short than multiple processes will try to access the SMS Inbox at the same time and the system will malfunction. If, on the other hand, the interval is too long than the SMS Inbox will remain unused for some time interval between the creation of a new process and the destruction of the last process. Such wastage of time also degrades the system performance.

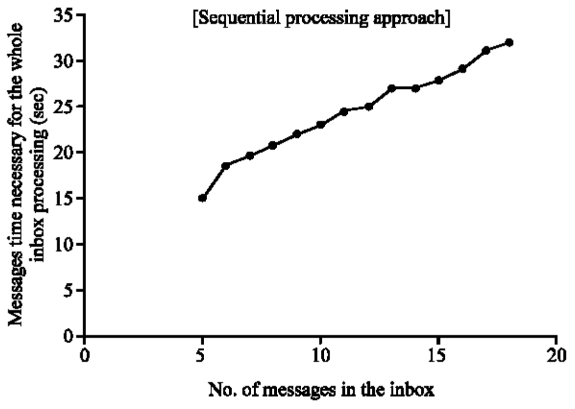

Fig. 5: Sequential approach

PERFORMANCE ANALYSIS

To analyze the performance of the system, the system was tested with 16 clients. All the three aforementioned approaches were tested by sending SMS messages from client mobiles to the mobile interfaced with the server. The basis of performance measurement was how many messages were processed by the system at some unit of time. To do that, the authors of the study measured the time it takes to process all the SMS messages in the SMS Inbox of the mobile set interfaced with the server. In the case of Multiprocess Approach Using Time Delay, different time delays were used in testing the system to find out single message retrieval. The observed data were used to produce some graphs. In this study such statistical analysis is presented.

Sequential approach: The algorithm of sequential approach (Algorithm 1) was tested by sending different numbers (5-20) of SMS messages to the gateway mobile interfaced. As shown in Fig. 5, the average time necessary for whole SMS Inbox processing is almost proportional with the total number of messages in the SMS Inbox. In the graph, total number of messages in the SMS Inbox is given in the horizontal axis and the time required for processing of these messages is given in the vertical axis. The graph, that is being drawn here, is not a straight line, partially because the varying arrival time of SMS messages. If the SMS comes from another mobile of the same operator, it arrives soon after it is sent; however, if the SMS is from a different operator company's mobile there is a little delay between the sending time and when the SMS is received. Later discussion of this study shows that this approach is slow compared to the multiprocess ones.

Multipreess approach using semaphore: The algorithm of multiprecess approach using semaphore (Algorithm 2)

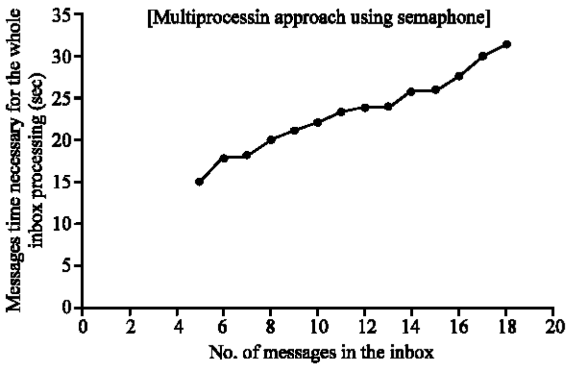

Fig. 6: Multiprocess approach using semaphore

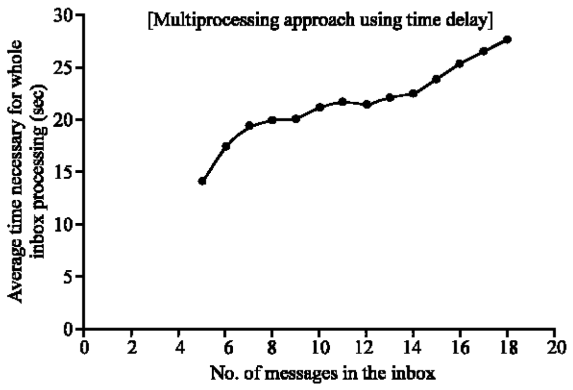

Fig. 8: Multiprocess approach using time delay

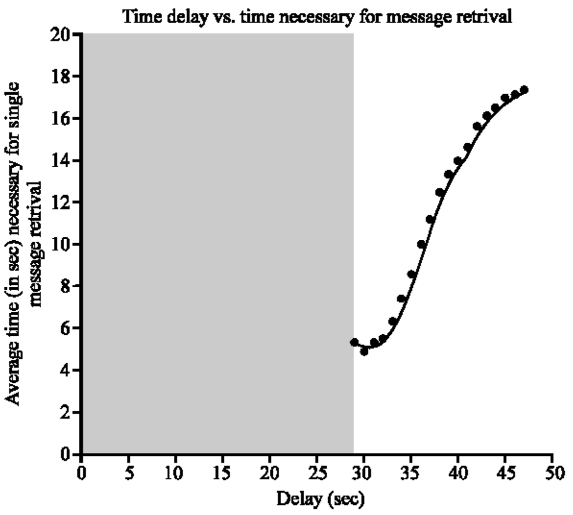

Fig. 7: Optimizing the delay for reliable retrieval of a single message

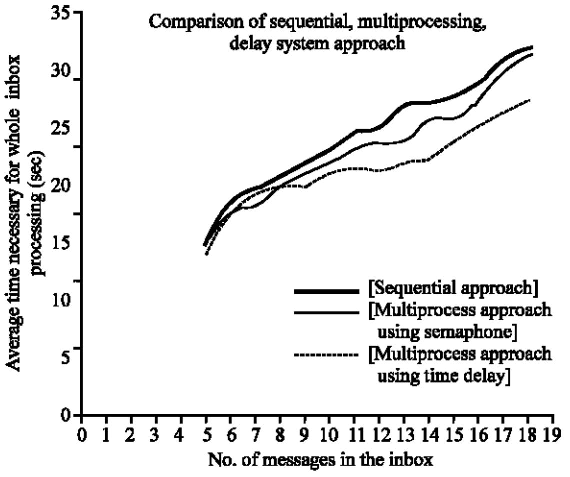

Fig. 9: Comparison between approaches

was tested in the very same way as done in this study. In the graph of Fig. 6, the total number of messages in the SMS Inbox is given in the horizontal axis and the time required for processing of these messages is given in the vertical axis. Like as the previous algorithm, in this case also, the average time necessary for whole SMS Inbox processing is almost proportional with the total number of messages in the SMS Inbox. It has been observed that for using multiple processes and using a semaphore to make the processes mutually exclusive, the overall average time to process SMS Inbox has slightly improved.

Multiprecess approach using time delay: The most critical challenge in using a delay so that multiple processes don't try to access the SMS Inbox at the same time, between 2 process creation is that the delay must not be too short so that there is always more than one process trying to access the SMS Inbox, on one hand. On the other, the delay must not be too long that the system sits idle, i.e. there is a time gap when no process tries to access the SMS Inbox. To find such an appropriate delay and to resolve this tradeoff, the algorithm of this approach has been tested with different delays (ranging from 10-50 sec) and the time necessary for single message retrieval were measured. The result is plotted in Fig. 7.

The grey-shaded area of Fig. 7 shows is from 0-29 sec. delay. In this region no observations about time could be made because, in such cases more than one process tries to access the SMS Inbox at the same time and the system gets hanged. It can also observed that 30.5 sec delay results in the minimum time for processing a single SMS. After crossing 30.5 in the vertical access, the time required to process a single SMS increases sharply with the delay.

The graph of Fig. 8 shows the time necessary for processing the whole SMS Inbox with respect to the number of messages in the SMS Inbox using Algorithm 3. The result is also similar. However, the overall time for whole SMS Inbox processing has been even decreased.

Figure 9 presents the observations of Fig. 5, 6 and 8 altogether to do a comparative analysis. It is evident from the figure that sequential approach takes the longest time for whole SMS Inbox processing, the Multiprocess Approach Using Semaphore works little better. The Multiprocess Approach Using Time Delay works almost the same when there is lower number of messages in the SMS Inbox, however, it works better when there is higher number of message in the SMS Inbox.

CONCLUSION

It is clearly evident from the statistical performance analysis of three approaches that when the total number of message in the SMS Inbox is low, the performances of all three algorithms are almost close. The performance varies a little when there are more or less 10 (around 50% of the SMS Inbox) messages. The performance of Multiprecess Approach Using Time Delay with a delay of 30.5 sec is significantly better when the SMS Inbox is almost full. Hence, it may be concluded that Multiprecess Approach Using Time Delay is the best amongst the prescribed approaches when the time delay is optimized at 30.5 sec.

This analysis, however, has some limitations. The system was tested sending SMS messages. The number of SMS messages was not symmetric for all three approach testing. Since no sufficient fund was available the project could not be tested for at every no of SMS messages in the SMS Inbox (0~20). In this study 4.3, while testing the optimal delay, 50% of the SMS Inbox (10 SMS) was filled up, the situation when there are more (>10) messages and the situation when there are less (<10) messages has not been tested. Although the system is designed for supporting official clients, but it could not be tested for such large number of doors.

To further improve the performance of the system, the number of gateway mobile phone sets interfaced with the server could be increased. It may be even better if these multiple mobile sets (interfaced with the server) contain SIMs of different operator company. This will reduce the air time of SMS message transmission. Instead of Nokia 3310, which is interfaced via serial port, USB or Bluetooth enabled mobile sets may be used and thereby the inherent speed limitation of accessing the mobile phone through serial interface may be eliminated.

REFERENCES

Albus, J.S., H.G. McCain and R. Lumia, 1989. NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), Technical Note 1235, National Institute of Standards and Technology, Gaithersburg, Md.

Björn, W., K.J. Åström and K.E. Årzén, Computer control: An overview. IFAC

Burns, A. and A. Williams, 1997. Real Time Systems and Programming Languages. 2nd Edn. Harlow, UK., (Chapter 3).

Cechticky, V. and A. Pasetti, 2003. Generative programming for space applications, In Proceeding of Data system in Aerospace (DASIA) Conference, Prague. Czech Republic.

Cechticky, V., A. Pasetti and W. Schaufelberger, 2002. A new approach to software development for embedded control systems. In Proceeding of MSy, Winterhur, Switzerland, pp: 75-83.

David, B., Stewart, Richard and A. Volpe, 1997. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects. IEEE. Trans. Software Eng., 23: 759-776.

Forum Nokia: Developer Resources. [Online] URL: http://www.forum.nokia.com/main.html.

GSM: Communicating with Nokia 3310 using data cable problem; www.experts-exchange.com/Programming/wireless_programming/gsm.

Hassan, H., R. Martnez, J. Sim and A. Crespo, 2002. Framework for developing real-time mobile robotic applications based on behavioural models. In 15th Triennial World Congress of the IFAC Barcelona. IFAC Publisher.

Khaled, S.M. and R. Karim, 2006. A Proposed Low-cost Security System Based on Embedded Internet Control. In Proc. SICE-ICASE International Joint Conference, Busan, Korea, pp: 4306-4311.

Khan, M.A.H., S. Saha, S.M. Khaled, M.T. Islam and M.S. Karim, 2007. Extending Mobile Phone SMS Capability into Email. Dhaka University. J. Sci., 55: 167-170.

Leslie, R.A., D.W. Geyer, K. Cunningham, P.C. Glaab, P.S. Kenney and M.M. Madden, 1998. Lasrs++ - an object-oriented framework for real-time simulation of aircraft, [Online] URL: http://jsbsim.sourceforge.net/fslinks.html.

Lloyd, B. and M. Susnik, 2002. Web embedded field devices. Pulp and Paper Ind. Technical Conf., pp: 199-202.

Musa, J.D., A. Ianniono and K. Okumoto, 1987. Engineering and Managing Software with Reliability Measures, McGraw-Hill, pp: 23.

Painton, L. and J. Campbell, 1995. Genetic algorithms in optimization of system reliability. IEEE. Trans. Reliability, 44: 172-178.

Pasettie, A.A., 2002. An object-oriented component-based framework for onboard software. In Proceeding of the Data systems in Aerospace Conference, Nice, France.

Sommerville, I., 2003. Software Engineering. 6th Edn. Pearson Education (Singapore) Ltd., pp: 286.

Wayne Peacock, 'Do you wish to control appliances from your mobile?' Copyright 2001-2005, http://www.embedtronics.com/nokia/fbus.html.

Xin F., S.A. Velinsky and D. Hong, 2002. Integrating Embedded PC and Internet Technologies for Real-Time Control and Imaging. IEEE/ASME. Trans. Mechatronics, 7: 52-60.