IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011
ISSN (Online): 1694-0814
www.IJCSI.org

A Remote Robotic Laboratory Experiment Platform with Error Management

Chadi Riman[1]

[1]**Computer Engineering, Fahad Bin Sultan University, Tabuk, KSA**

Abstract

Remote control of experiments is gaining more importance in training and education. However, remote real-time training on instruments programming still have some unresolved problems such as error management.

In this paper, a platform for training students on system's control by Tele-Programming is presented. Programming sessions can be done by the trainee at many levels of control with built-in error management in order to avoid system freezing or malfunction. We showed an illustrative application: programming navigation control of a mobile robot in the presence of obstacles using fuzzy control.

Keywords: *Remote lab experimentation, HCI, Robotics, Computer Simulation.*

1. Introduction

Remote Experimentation is a distant control of an experimental setup accessed from different places and by different users (figure 1). The application carried out in Remote Experimentation can vary from a simple demonstration where interaction between the student and the experiment is on a simple level (view only), to a complex application where the student has more control over the experiment. The first case is safe with limited teaching possibilities. The complex case has more teaching advantages but it also has malfunction risks due to a higher probability in committing errors by students. This type of training can be improved if it is accompanied with a tutorial and simulation software to be used before the real experiment.

Tele-Programming is a remote laboratory platform in which control is done using program files exchange. These files are usually text files of small size which requires very low bandwidth. This type of remote experimentation is therefore suitable to low speed networks. This study evaluates some existing major platforms in tele-programming and suggests an improved low-cost platform with three programming levels based on student and course levels. For illustration purposes, this platform will be used for remote training on a mobile robot in a fuzzy logic environment.

The main problem in self programming is the need of a tutor either in the local place or in the remote lab, which can be replaced by a tele-tutorial system [1-3]. This is also achieved in our platform by an error management module to identify errors, notify the student, and prevent system malfunction. Our idea is to support the training of the students by allowing failures in the experimentation. The system can manage different kind of failures and then send feedback to student. Moreover, all processes are built using free software.

In this paper, section 2 presents an analysis of some existing tele-experimentation training platforms used in robotics. The need of different programming levels in training is discussed in section 3. The suggested platform architecture is given in section 4. Section 5 presents the robot and its fuzzy controller module which is used for training in our suggested platform. Error managements and Simulation modules are respectively described in sections 6 and 7. A case study is given in section 8 and concluding remarks are given in section 9.

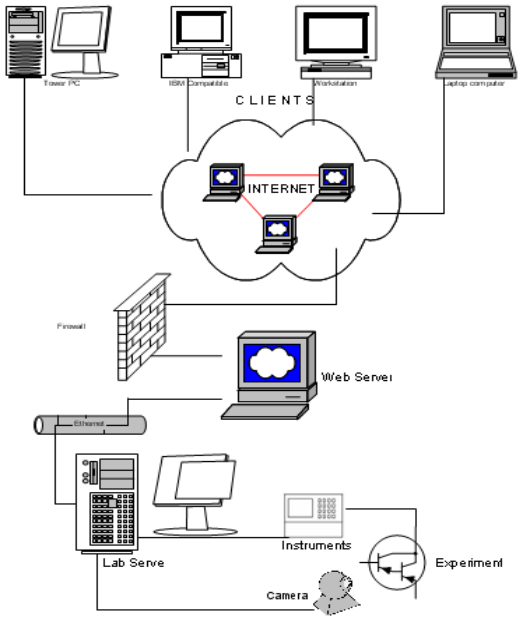

Figure 1. Remote Experimentation

2. Existing Robotics Platforms

Distance Learning platforms on robotics (DLR) are mainly concerned with learning, evaluation and security. In this section, the characteristics of some Existing DLR platforms [4-7] will be studied and used in the analysis of our suggested one.

The Open Learning platform presented in [4] uses MS NetMeeting and Matlab real-time tools for control purposes. It has the following characteristics:

- Software development is simple, reducing expenses and minimizing faculty work.
- Existing off-the-shelves freeware software, are used.
- No need for Web-enabled Interfaces

This platform uses the *Learning-by-Doing* methodology but it doesn't present any solution related to safety problems.

The platform described in [5] uses the *Active-Learning* methodology. The design provides the remote user with the perception of reality which is due to the use of Learning Objects. Safety is provided by running a VRML (Virtual Reality Module Language) simulation before executing the control program, which may reduce any damage due to some manipulation errors. Control learning is limited to changing pre-defined controllers with their parameters.

In the Platform described in [6] uses the *Learning by Tele presence*, the *Learning-by-Doing* or *Active-Learning*. The remote user indicates obstacles to be avoided and the target to be reached. A simulation module based on a potential field algorithm draws the path and a control program runs in order to follow it. Infrared sensors are installed on the robot to provide security and increase autonomy. This system has some deficiencies due to the limited interaction with the user and limited experience of the student.

The platform developed in [7] uses all learning methodologies previously mentioned for remote control of Lego mobile robots. The student uses Matlab/Simulink in order to design a controller to track a user defined trajectory. The control program is next transmitted to the server and executed. Three lights have been placed on the top of the robot in order to detect position and direction by means of a camera. A safety mechanism stops the experiment whenever the robot reaches a forbidden region. The control accuracy is based on a predefined model of the robot dynamics and needs to be changed with the physical environment.

The platform suggested in our work focuses on displacement control of a mobile robot. It benefits from useful techniques and methodologies developed in previous studies and add improvements to them.

3. Programming Protocol

The suggested protocol for training on programming shows that there are three levels of tele-programming (Figure 2):

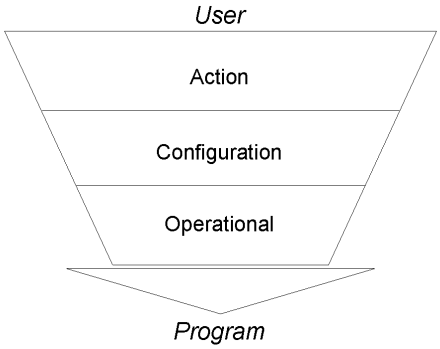

Figure 2. The three programming levels

1) **The direct action level** is for introductory courses where a program corresponds to a sequence of instructions (advance, turn…). The first set of experiments is made at this level where the student will be learning the system specifications (functionalities and workspace) and the basic programming structures (loops, iterations…).
2) **The configuration level** is used for more specialized courses where programming integrates internal specifications. The student can modify internal parameters. This facilitates the understanding of control principles.
3) **At the operational level** the student learns how to control system by advanced programming. The program file is transmitted for execution on the server. The program can be directly executed either on the robot or its virtual simulated model. Virtual reality, for design engineer workshop, allows transmitting control instructions without syntax constraints. Contrarily, a textual way for transmission of control instructions needs more abstraction in the programming phase.

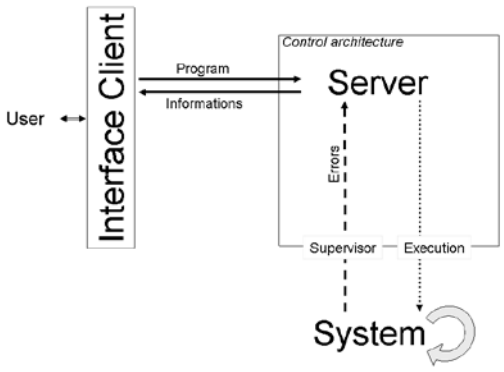

Figure 3. Process of failure supervision

An error management module (Figure 3) is included to deal with failures produced at any of the described levels. This module prevents deadlocks [8] and system freezing, and transmits feedback information which improves the student learning. It runs on the server in order to validate syntax and semantics of programs. It also supervises program execution like in a watchdog concept. In addition, it controls the execution time and odometrical task limits. Therefore, students can test their programs without human tutor because of the feedback provided due to an abnormal response of the robot.

4. Architecture of Suggested Platform

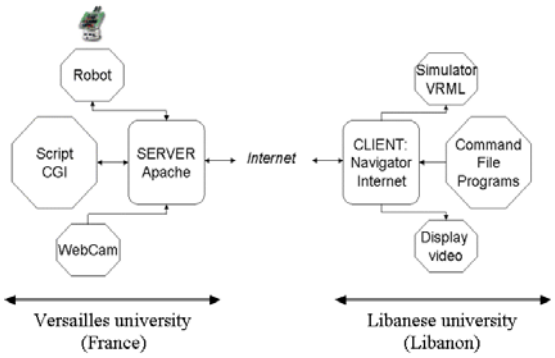

Figure 4. The platform architecture

The suggested platform uses client-server architecture (figure 4). The server provides training information such as description of programming methods, programming steps, and programmable devices. It is also responsible of communication with the remote client and with the robot.
Interoperability among users is achieved by using an Apache server and an HTTP browser. Apache server is very stable and widely available. The server software includes the modes of displacements, the reachable workspace of the robot, the response times of the actuators and the sensors as well as the sequence of procedures required for a given task. Interactions between the user interface and the platform is based on sending predefined instructions to the robot, and receiving its status (Figure 5).
This interaction is performed in order to explore the robot parameters and them to program its operation. Exploring parameters allows the understanding and the comparison between structures and sensitivity ranges of various controllers. Programming provides the ability of integrating the controller in a programming language. The program has to follow a predefined structure in order to be compatible with the error management module. In case of errors, the system sends an error report and reinitializes the platform for restarting the exercise.

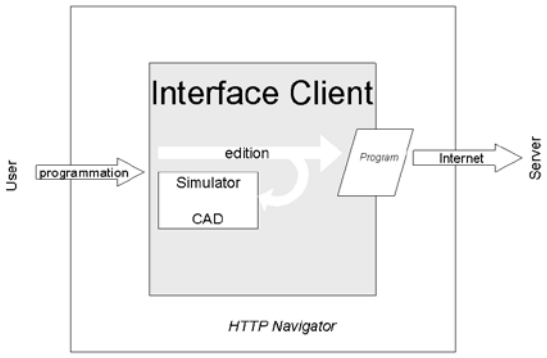

Figure 5. Interface process

The Apache server communicates with the robot through a Common Gateway Interface (CGI) using C language. The user program is transmitted by the server to the CGI which transmits information to the robot according to the programming level. The robot is connected to the server through an RS232 HF serial port. Next, the CGI Program returns results to the client. A Webcam connected to the site returns feedback information on the framework environment.

5. The Robot and its Fuzzy Controller

The Khepera robot (Figure 6) has a diameter of 55 mm and a height of 30 mm. It is controlled by a Motorola processor 68331 with 256 KB of RAM and 18 KB of ROM. Its motion is due to two DC motors with encoders. It is also provided with 8 infra-red sensors. Because of its modularity and its important number of options, this product is widely used by researchers and teachers. It can be programmed in GNU C with LabView or Matlab.

Figure 6. Khepera mobile robot

At the first level of programming, operations are based on actions available in the robot integrated libraries. These actions are simple: advance, turn, pause, avoid, measure sensors values... The user is therefore able to carry out a simple task in a complex environment.
At the configuration level, uploaded programs use fuzzy controllers that are structured with heuristic features close to human actions. Configuration is done on parameters relative to a classification of

entries (data generated by sensors or robot status) and on fuzzy rules.
At the operational level, the transmitted source code implements the algorithmic structure and the robot control.

6. Error Management

For a navigation task towards a goal in a complex space, two situations are considered as [8]:

- Blocking (figure 7): a bad choice of control parameters (control law or range of operation of proximity sensors) may cause a freezing in operation during obstacles crossing.
- Vagrancy (figure 8): the error causes divergence from the goal.

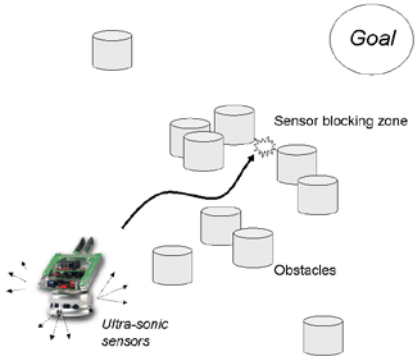

Figure 7: blocking situation

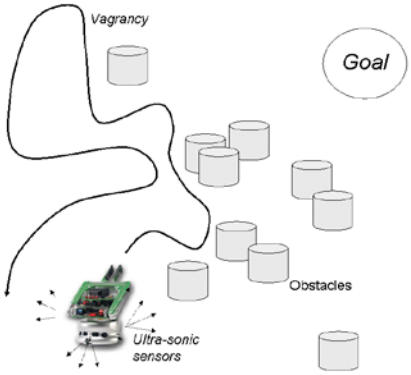

Figure 8. Vagrancy situation

A first type of errors is due to a bad parameters configuration. These fuzzification parameters on inputs and on selected rules of inference, lead to freezing during a simple navigation task in presence of obstacles. This allows students to understand the structural and logical definition of the configured fuzzy subsets. A simple modification of the number of subsets or the use of a symmetrical or asymmetrical triangular structure makes a navigation task successful or not.

The second type of errors occurs because of a programming problem (infinite loops, interruption, exception...). For example, infinite loops are suspected when blocking occurs without being associated with sensors configuration. Memory allocation problem is detected by the mechanism of integrated watchdog: the robot must send periodically information on the execution status. The training at this stage is concerned with the algorithm, its implementation, and its execution. Tools for coding and reliability analysis could be used during this learning stage [9].

7. Simulation Module

The aim of the simulation module is to perform a local test on the student program before being uploaded. The simulation phase has the following objectives:
1- Understanding the robot specifications and functions.
2- Testing programs without risks or without using the robot.
Simulation is done in a VRML (Virtual Reality Modeling Language) environment, which is a 3-dimentional scene description language installed on the client station. First, the server transmits various documents to the client browser which interprets these documents with a possible help of plug-ins. In case this interpretation fails, these documents are transmitted to the VRML.

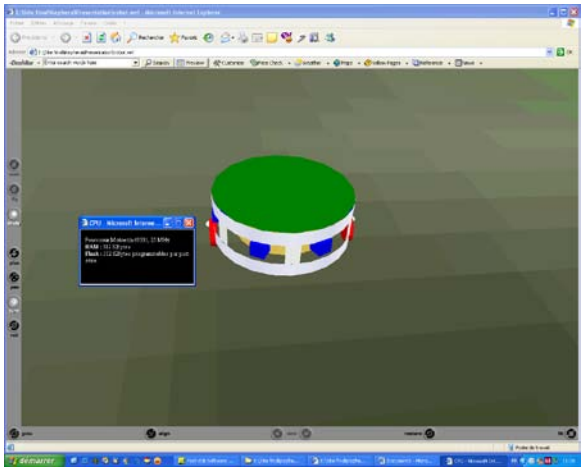

Figure 9. VRML Simulator

In a first step, the simulator visualizes the robot in order to present its functionalities (figure 9): motion, perception and processing devices. Students, at this stage of training, can study these functions relative to mechanical, electronic and data processing concepts:

- Perception: Allows studying the principles, types, ranges, and positioning measurements of infra red sensors.
- Motion: Allows to study actuators specifications and types, as well as kinematics and power module devices.

- Processing: Allows studying processor characteristics (memory, temporal diagrams...).

In a second step, VRML simulator serves as a trial stage for programming. Students can study the behavior of the simulated robot in the action programming level.

8. Case Study

Experiments were carried out between Lebanon and France for various levels of programming. This allows evaluating the risks of operation due to the reduced quality of connection. The system was installed on a computer server at the LISV lab (Versailles University). The server was connected by wireless link to the robot. A visual feedback was used during the experiment in order to simplify operations and to improve learning process.
The test task was a simple navigation with obstacle avoidance. The Khepera mobile robot was installed in a limited enclosure for a safe displacement. Examples are given for various types of users (specialized engineers, students without or with limited experience) operating at different programming levels.

a. Direct Action level
This level is tested with high school students. They have to program the robot in order to test sensors and control strategies based on program files composed of predefined orders. A list of commands accessible from the client site in action mode is given in Appendix.
Table 1 summarizes the common errors which are mainly due to bad values of PID parameters or other parameters relative to uploaded actions. Errors are always detected by means of mobility status.

Cause	Effect
Bad choice of PID terms, Often Integral term too high	*Instability and Vagrancy situation*
Displacement without sensors feedback	*Blocking situation because of security process occurrence*

Table 1. Common Students errors in the Task level

b. Configuration level
At this level, a student tests his/her knowledge of control based on fuzzy logic. Thus, to simplify illustration of obstacle avoidance management, outputs of sensors will be fuzzified in terms of detected distance and orientation. An example in figure 10 can be adapted by modifying the structure of the fuzzy subsets: i.e. by modifying each subset limits FSij. This information will modify the mobile robot behavior by affecting its sensitivity to sensors values. In the same way, it is possible to modify the rules of fuzzy inferences. This technique is based on the Sugeno-takagi approach [10], in which conclusions of the rules are singletons Si (Figure 11). The file of the program on the configuration level contains the parameters FSij and Si. Table 2 summarizes common errors that may occur at this level.

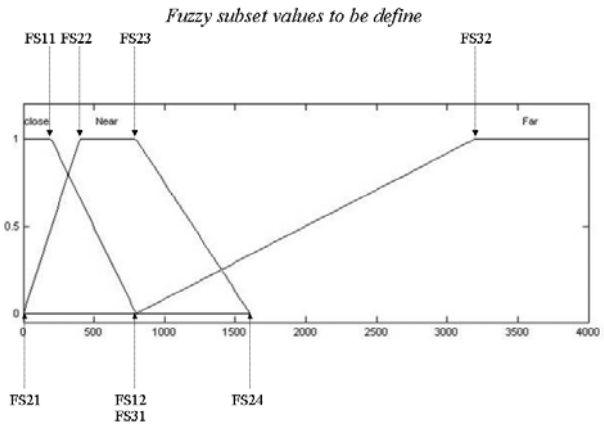

Figure 10. Example of fuzzy sub sets

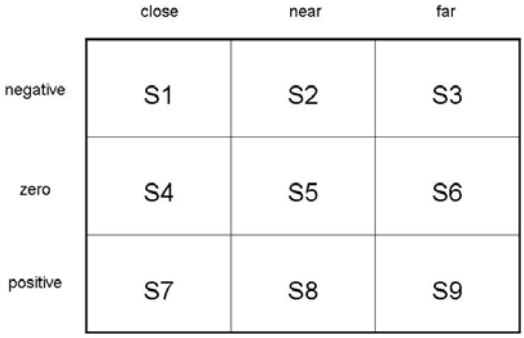

Figure 11. Inference rule table with singletons

Cause	Effect
FSij terms shift on the left space (obstacle detection too close)	*Blocking situation*
FSij terms shift on the right space (far obstacle detected)	*Vagrancy situation*
Si terms in the inference rule table are symmetric	*Blocking situation by opposition between two obstacles*

Table 2. Common Students errors in the Configuration level

c. Operational level
At this level, students must carry out the complete compilation in order to upload the program file. To simplify this phase, students may use a library of programs made of classical Khepera instructions and stored in the server. It is possible to proceed to simple actions like flickering LEDs of the robot, reading values of the infra-red sensors or writing a program containing navigation instructions with

obstacle avoidance such as the one based on the principle of Braitenberg [11]. Uploaded files in this level must respect the S37 format [12]. The robot being provided with a microcontroller Motorola 68331, programs written in C language, must be compiled using an appropriate cross-compiler like KTproject compiler under Windows. In this mode, the program is automatically executed after it is uploaded on the robot.
Table 3 contains common errors that may occur at this level.

Cause	Effect
Braitenberg simple control	*Blocking situation by antagonism between two obstacles*
Bad choice of control law	*Vagrancy or blocking situations*
Infinite loop problem	*Hazardous direction vagrancy situation*
Memory allocation problem	*On place blocking situation*

Table 3. Common Students errors in the Operational level

9. Conclusion

In this study, a platform for training on systems control by tele-programming was presented. According to the student academic profile, and in order to improve self-learning process, a protocol including various training levels and error management was validated.
This approach was evaluated based on experiments carried out through Internet connection between Lebanon and France. The experiment for the student learning phase was a mobile robot programming using different levels: from predefined order to fuzzy logic programming. During this stage, we tested the error management to improve knowledge feedback for students.
The realized platform uses free software and works with low bandwidth Internet connection. Our idea is to answer typical student constraints in term of flexibility and cost.

After this first trial supported by CEDRE program[1], different modes of operation will be introduced in a future work. A suggested mode of operation is to configure the system to be used by handicapped persons. For this purpose, the learning platform would integrate an adaptable and easy configurable man machine interface.

[1] "Coopération pour l'Évaluation et le Développement de la Recherche", Cooperative program between Lebanese and French Governments.

Appendix

List of commands accessible from the client site in action mode

A: for parameters configuration of the PID velocity controller: proportional (K_p), integral (K_i) and derivative (K_d).
Default values of these parameters are: K_p=3800, K_i=800, K_d=100.
C: Indicates to the position controller the absolute position to reach. The robot trajectory will produce three phases: acceleration, constant velocity and braking.
D: Configures the velocity of both wheels. The unit is pulse/10ms which corresponds to a velocity of 8mm/s, its maximum value is 1m/s.
E: Reads the instantaneous wheels velocity.
H: Reads the position 32 bits counter of each wheel.
I: Reads on 10 bits the value of the analog input relative to the selected channel. Its maximum digital value corresponds to an analog input of 4.09 Volts.
Channel 0: Detects battery status.
Channel 1: Measures instantaneously the intensity of the reflected light.
Channel 2: Measures instantaneously the intensity of the ambient light.
Channels 3, 4, 5: Free channels to be used by analog inputs: 36, 37 and 38 of the KBus.
Channel 6: Reads the Khepera current consumption in mA.

J: Configures velocity profiles using motion parameters (Maximum velocities and accelerations).
N: Reads on 10 bits each value of all eight proximity sensors.
P: Configuration of the desired amplitude of the PWM relative to each wheel. The modulation factor varies between 100% lagging and 100% leading with 0% as middle range. These values correspond respectively to +255, -255 and 0 as binary reading.

Acknowledgments

The author thanks the Assistance and Handicap team headed by Dr. Eric Monacelli in LISV research laboratory of Versailles University (France) for their help in applying this work.

References

[1] A. Böhn, K. Rütter, B. Wagner, "*Evaluation of tele-tutorial in a remote programming laboratory*", American society for engineering education annual conference, 2004
[2] C. Riman, A. El Hajj and I. Mougharbel. "*A Remote Lab Experiments Improved Model*", International Journal of Online Engineering, Volume 7 Number 1, 2011.

[3] I. Mougharbel, A. El Hajj, H. Artail, and C. Riman. *Remote Lab Experiments Models: A Comparative Study*, International Journal of Engineering Education, Volume 22 Number 4, 2006.
[4] N. Swamy, O. Kuljaca, F.L. Lewis, "*Internet-Based Educational Control Systems Lab Using NetMeeting*", IEEE Transactions on Education, VOL. 45, N°. 2, May 2002.
[5] D. Fabri, C. Falsetti, S. Ramazzotti, T. Leo, "*Robot Control Designer Education on the Web*", Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, April 2004.
[6] F.D. Von Borstel, B.A.. Ponce, J.L. Gordillo, "*Mobile Robotics Virtual Laboratory Over the Internet*", Proceedings of the Fourth Mexican International Conference on Computer Science (ENC'03), 2003.
[7] F. Carusi, M. Casini, D. Hattichizzo, A. Vicino, "*Distance Learning in Robotics and Automation by Remote Control of Lego Mobile Robots*", Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, April 2004.
[8] A. Smirnov, E. Monacelli, S. Delaplace "*Single adaptation mechanism for collaborating multi-robots*", ICINS'2002 conference, St Petersburg, Russia, 2002.
[9] M. H. Klein, and al., "*A Practitioners' Handbook for Real-Time Analysis: Guide for Real-Time Systems*", Boston, Kluwer Academic Publishers, 1993.
[10] F. Abdessemed, K. Benmahammed, E. Monacelli, "*A fuzzy based reactive controller for a non holonomic mobile robot*", Robotic and autonomous systems journal, pp 31-46, 2004
[11] V. Braitenberg, "*Vehicles: Experiments in Synthetic Psychology*", MIT Press, 1984
[12] K-Team, "*User's Guide for Khepera mobile robot*", http://www.k-team.com/download/khepera.html

Chadi Riman received his Bachelor of Engineering and Masters of Engineering degrees both in Computer & Communication Engineering from the American University of Beirut (AUB), Lebanon in 1994 and 2004, respectively. He finished his PhD degree at the University of Versailles (UVSQ), France in January 2008. He worked from 1994 to 1998 in the software engineering domain, and was the IT manager in AinWazein Hospital, Lebanon from 1999 to 2008. He is currently Assistant Professor at Fahad Bin Sultan University, Tabuk, KSA. His research interests include software systems for handicap rehabilitation and remote engineering education.