MINDLAB, A WEB-ACCESSIBLE LABORATORY FOR ADAPTIVE E-EDUCATIONAL ROBOT TELEOPERATION

P. Di Giamberardino, M. Spanò Cuomo and M. Temperini
Department of Computer and System Sciences, University of Rome "La Sapienza", Rome, Italy

Keywords: WebLab, e-learning, Robot control, Teleoperation, Lego Mindstorms.

Abstract: The paper deals with a web based remote access laboratory, called MindLab, for educational support, via e-teaching and e-learning approaches, in practical robotic experiment to students of robotics and control theory courses. The hardware components are based on commercial easy to use, cheap and modular Lego Mindstorms devices. The software part, in addiction to the framework required for web access to the hardware resources, is designed and implemented in order to allow the teacher to define particular didactic paths depending on each student's skills and on the desired final abilities to be acquired. Exercises are proposed following a skill/competence level set, enabling higher levels access according to the results obtained. After a short description of the hardware setup, the software functionalities, mainly focusing on the exercise definition and presentation aspects, are illustrated, giving also an example of a standard exercise. Concluding remarks end the paper.

1 INTRODUCTION

A crucial aspect in both face to face and distance learning is in the need for direct hands-on learning experiences, conducted by the students in laboratory (Traylor R.L., Heer D. and Fiez T.S., 2003; Gomes L. and Bogosyan S., 2009).

Unfortunately, there are severe limitations in the availability of sufficiently large and equipped laboratories to cover an access reasonably open to all the needing students.

A remote environment would allow the students to operate their experiments in distance, by managing their queue and supporting the learning experience. The effect is a more efficient sharing of the available resources and a potentially full time availability. This advantage is often stressed in several applications (Gomes L. and Bogosyan S., 2009; López D., Cedazo R., Sánchez F.M. and Sebastian J.M., 2009).

Implementing the networked solution by a web-based application does have its merits (Gomes L. and Bogosyan S., 2009; López D., Cedazo R., Sánchez F.M. and Sebastian J.M., 2009; Balestrino A., Caiti A. and Crisostomi E., 2009), in that it allows easily widespread availability of the exercise environment: the "communication environment" is basically a web-browser; students can operate from home, directly; no problem of installation of proprietary software on students PC should occur.

Such support could be in the visualization of experiment results, and in the provision of data for error analysis; and it can be also less conventional, and extend to recording and personalizing the student activity (e.g. by adapting to the learner's experience and goals the list of accessible exercises).

We propose a web based system interfacing robots for laboratory web-based activities. It supports learning activities in planning and control of autonomous robotic devices; different robots, in different configurations, can be available (either at the same time, in differently dressed areas of the laboratory, or in series); the exercises are defined by the teacher and specified to fit for certain learning goals and to certain robot configurations: learning goals of an exercise are specified in terms of difficulty to meet and competence to gain; *paths of experiences* are defined by the teacher as a sequence of exercises that the student is supposed to be able to perform at the end of the learning activity in the course; each learner is associated to a profile that is maintained during her/his activity: exercises are listed as available to the given learner only if the her/his profile according to the exercise's learning goals; after an exercise has been completed by the learner, her/his profile is updated by the skills gained from the learning goals and new exercises in the path are possibly made available.

Di Giamberardino P., Spanò Cuomo M. and Temperini M. (2010).
MINDLAB, A WEB-ACCESSIBLE LABORATORY FOR ADAPTIVE E-EDUCATIONAL ROBOT TELEOPERATION.
In *Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics*, pages 309-314
DOI: 10.5220/0002953303090314

The laboratory is located at the Faculty of Information Engineering of Sapienza University, in the premises of Latina, and it can be reached at the web address http://infocli31.dislt.uniroma1.it/webRobot/, where at present, due to its current use, only a Italian language version is available.

The main contribution is related to the particular software design which gives the possibility of specific didactic paths construction, tailored on each student initial skills and desired learning goals.

In the most recent literature, several examples of web based laboratories are illustrated. As evidenced in (García–Zubia J., Orduña P., López–de–Ipiña D. and Alves G.R., 2009), the most of them put their main attention in the hardware aspects, banishing the software to a little more than a mere communication interface.

A wide analysis of the state of the art in remote laboratories is performed in (Gomes L. and Bogosyan S., 2009), where a first simplified classification of local/remote and simulated/hands-on experiments is given, and some benefits are evidenced.

Several applications of virtual and augmented reality have been known to have some didactic usage, such as in (Kosuge K., Kikuchi J. and Takeo K., 2002; Marín R., Sanz P.J., Nebot P. and Wirz R., 2005). An application of remote accessibility of an experimental lab for student with disabilities is in (Colwell C., Scanlon E. and Cooper M., 2002). iLab (Lerman S. and del Alamo J., 2000-2005) makes accessible laboratories developed at MIT.

Regarding the interaction between remotely available laboratories and distance learning services, (Chellali R., Dumas C., Mollet N. and Subileau G., 2009) laments that the problems, in making complex systems available through e-learning, are rather conceptual than technical.

(Borgolte U., 2009) presents the architecture of a remotely operable laboratory with a mobile robot, used for education. Programming is done in COLBERT, offering quasi-parallel execution of activities.

In (Casini M., Prattichizzo D. and Vicino A., 2004) Automatic Control Telelab is presented, as a web-based (through java applet technology) system, allowing to put on-line a series of experiences in various fields of automation and control. Students have access, through the "experiment interface", to the graphically and video rendered outputs of the experiment. There are several experimental settings, and an additional field related to LEGO Mindstorm is announced.

A survey on Web technologies used in control systems courses can be found in (Poindexter S. E. and Heck B. S., 1999), where the authors describe the use of virtual (Merrick C. M. and Ponton J. W., 1996; Schmid C., 1998) and remote labs. A distinguishing feature of remote labs as compared to virtual labs is that users can interact with real physical processes through the Internet, making them more attractive than controlling software simulations. At the same time, the design and implementation of a remote lab is more challenging due to safety and fault tolerant aspects.

In (Jara C. A., Candelas F. A. and Torres F., 2008) an environment for remote operation and simulation for a robot arm is described. It uses easy java simulation for the applet-based graphical interface, making it necessary on the local PC only a web browser and java+java3D runtime system. A dedicated and expensive robot is made available.

(López D., Cedazo R., Sánchez F.M. and Sebastian J.M., 2009) describes a web accessible experimental setup for embedded real time systems programming for a robot arm. Free software is used both for client and for server applications. The hardware setup is fixed and the different sessions propose different tasks to be executed by the robot arm; the tested skill is the ability of produce a correct and functional program for executing the given task.

The here presented remote laboratory, MindLab, has been designed and built up focusing on the possibility of a quite flexible and configurable exercise settings, where three main components - configurations of robots, experiment areas, exercises definitions - of the remote learning activities can be accommodated in varied manners: different robot(s) configurations can be defined, different playground(s) settings can be arranged and (clusters of) exercises can be defined according to specific robot and layout configurations.

Our system provides a quite simple-to-use environment supporting programming and experimenting on Lego Mindstorms robots, a family of commercial robots which is widely available, reasonably sophisticated in terms of functions, pretty easy to use and sufficiently affordable in terms of cost. Moreover, being intrinsically modular in terms of sensors and actuators number and distribution, is is easily and quickly reconfigurable and several different configurations can be obtained starting from the same set of devices.

In MindLab, though, the main issue we worked on so far is in giving rise to a truly e-learning framework, capable to support methodologically and pedagogically sound distance activities, and in particular personalized and adaptive management of the learner's experience (Fernandez, G., Sterbini, A. and Temperini, M., 2007; Limongelli, C., Sciarrone, F., Temperini, M., and Vaste, G., 2009; Sterbini, A. and Temperini, M., 2009; Limongelli, C., Sciarrone, F.,

Temperini, M., and Vaste, G., 2010).

2 THE MINDLAB FRAMEWORK

MindLab is basically a web-application running on a server resident into a small laboratory: the server can access, via bluetooth connection, the robots, as instructed by the application. The communication scheme is depicted in Figure 1

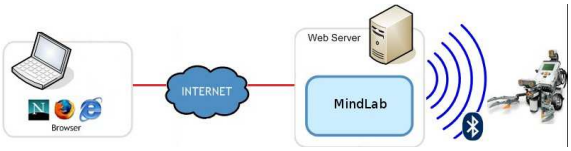

Figure 1: Communication scheme in MindLab.

The system allow to make exercises with robots based on the LEGO Mindstorms NXT technology. This choice has been driven by the idea that such a system could give sufficient flexibility to the laboratory experimental layouts, besides being economically feasible and allowing to use and maintain several robots.

To program the robot, the support can be based on graphical interfaces, such as in NXT-G, Robolab, and NI LabVIEW Toolkit, or on the use of a plain programming language, such as C, C++, URBI, leJOS, NXC, NBC. The original firmware included in the "brick" NXT does support only the last two languages; all the other systems need a dedicated firmware to replace the original.

Then, a first issue was in the possibility to replace the original firmware of the brick by a new one, more extended and complex. The reward in that replacement was in the possibility to get total control of the hardware, import algorithms (e.g. from artificial intelligence disciplines) more easily and enjoy better performances. A second issue for decisions was in the programming language to be supported by the system.

As of the former issue, we thought that full control could also make it necessary to display higher level of programming skills and to deal with less predictable and tractable programming errors.

Then, the decision to limit programming skills and freedom and maintain the original firmware was adopted.

Regarding the latter issue, another constraint in our project was the will to make an additional interface available to input pseudo-code, rather than plain programming code. The pseudo-code (that is a more straightforward way to specify the robot behavior, expressed through system supported menus of commands) would then be translated in the programming language supported by the system before processing its execution. This would allow for a lower/simpler interaction level between student and system: it can be useful at initial stages. At a higher level of student skills and responsibility, direct input of plain programming code is also possible. So one of the factors in our choice of the programming language supported by the system was in having it to be the target for pseudo-code translation. These considerations bring us to selected *Not Exactly C* (NXC). NXC is an high level programming language deriving from the NBC (*Next Bytes Code*, used in previous LEGO technology *RXC*)

Communication between server and Bricks can be based both on a cabled USB connection and on a wireless Bluetooth one.

To complete the hardware architecture, a webcam based system for the remote visualization of the running experiment is included.

3 THE EXERCISES ENVIRONMENT

The system can manage several *courses*; each course is an environment in which paths of exercises are defined, to reach determined skills.

An exercise is defined by specifying the robot layout for its execution and two cognitive aspects: difficulty degree and competence level (both are numbers in [1,5]; the latter represents the skills that are demonstrated/gained by solving the exercise).

Each course have several students enrolled, and one or more teachers responsible.

An administrator manage the system, e.g. grant to certain users teacher privileges, create new courses and assign their teachers, configure exercise assessment and execution ways, browse the log file containing records of the system activities, and configure in the system the robots that are available on the ground (basically asserting the physical robot configuration in terms of sensors and engines attached.

The students enrolled in a course have access to the list of exercises they can perform in that course, and to the records of performed exercises. A student can perform and exercise if the related difficulty and competence level are *sufficiently close* to her/his skills. A learner's skills are represented through a model, constituted directly by the records of the exercises previously solved by the learner.

To solve an exercise, the student devises a computer program (either directly, in NXC, or by listing its pseudo-code through and assistive interface), and,

after being entitled to access an available robot, inputs the program and sees the results. As a matter of facts, exercise assessment is just semi-automatic: the final judgment is stated by the teacher, that can declare the exercise solved to grant the learner with the related difficulty and competence levels.

Then, the teachers monitor and assess the exercises performed by their courses' students.

Moreover they specify new exercises. Figure 2 shows part of the definition interface. Here the less straightforward among the data managed by the system to define an exercise: ROBOT CONFIGURATION for the exercise, selected among those defined by the administrator, corresponding to the machine actually available in the lab; HARDWARE SPECIFICATION, i.e. the set of devices (sensors and actuators) that are supposed to be needed for the exercise (and so present in the robot configuration); DIFFICULTY degree, meant to express how challenging, in terms of coordination and complexity of the control, the endeavor can be; it's an integer number $\in [1, \ldots, 5]$; COMPETENCE level that is supposed to be shown in solving the exercise; it's a number as above. Note that this is not a prerequisite: it is more similar to the certification of a learner's *cognitive level* (cf. Bloom, 1964) granted by the learning activity associated to the exercise.

There is one more part in an exercise definition, and it is very important: the activity is definitely specified by listing a set of *objectives*, sort of sub-tasks for the whole activity. Although such segmentation could be pedagogically useful in providing an order of action for the students, it is not particularly stressed in the syste: As a matter of facts, it is over the objectives that the teacher, with the system's assistance, actually states the difficulty and competence levels, and the hardware specifications. Then, the exercise is the *summa* of the sub-tasks (its levels are the highest among the objectives' ones, and its hardware is the union of the objectives' requests).

The definition interface for objectives is in Figure 2.

Figure 2: Two objectives in the definition of an exercise.

To help teachers in assigning difficulty levels to objectives a table of correspondence between keywords and levels is available. For instance, keywords such as "Turn+DIRECTION" or "Stop" are given automatically difficulty 1; "Measure" is level 3; "Looking_For" is level 4. Presently we have 30 keyword (distributed in the 5 levels) in the table. The teacher is also provided with a set of accompanying recommendations related to the pedagogycal interpretation of the keywords/levels; e.g. if the objective requires to "manage different types of movements" it will have at least a classification 2 of difficulty.

Competence levels basically express programming capabilities and the knowledge and solution of typical problems. In order to mark such level in an objective the teacher is required to peruse an help table, specifying at the moment, some remark about each level and 20 indications on the levels, overall. For instance, basic knowledge of sensors and engines, basic terminology, basic movements, are the indications matchable in cognitive level 1; at level 4, instead, capabilities such as "knowledge of the hardware elements and ability to let it interact", "capability to use all the programming commands supported in MindLab"; while a level 5 capability is "knowledge of MindLab commands limits".

Keywords and recommendations are not fixed once and for all; we think that different courses (meaning exercises cycles) and teachers might require different descriptions for the levels of difficulty and competence. So, keywords and recommendations in the system are actually completely reconfigurable and expandable, also at the level of the single course (and related teacher). We think this can help avoiding to frame everybody in a predefined pedagogical format, and increase teacher comfort in the system.

3.1 An Example of Exercise

TEXT OF THE EXERCISE

The robot should follow a black line on the floor. At line end, it has to check around for obstacles closer than 15cm. If no obstacle is seen, emit a sound, otherwise measure the intensity of light reflected by the floor and output it.

OBJECTIVES

An analysis of the text (we underlined significant segments) helps defining the objectives of the exercise. They are enumerated in the following, giving *keyword*, *difficulty* (*D*) and *Competence level* (*C*): *Follow a Line*, $D = 3$; $C = 3$; *Check* (for obstacle), $D = 4$, $C = 4$; *Emit Sound*, $D = 1$, $C = 2$; *Measure*, $D = 3$, $C = 3$.

So the exercise has four objectives and a overall

qualification of: Difficulty = 4; Competence = 4.

3.2 Solving an Exercise

Figure 3 shows the student interface for exercises. Exercises whose solution the student can attempt are pointed out by a green dot. Those exercises for which the learner submitted a solution are also pointed out, to mean whether they have been accepted or rejected by the teacher, or their assessment is pending. Exercises that are defined with hardware requirements not matching an available robot configuration would be pointed out (not in figure).

Figure 3: The learner's interface to exercises. The first in list has $D = 1$ and $C = 1$. The second was attempted and is pending assessment. The others are not yet available since their Ds and Cs are not yet suffciently close to the leraner's profile. When the submitted exercise will be accepted, the profile will increase to $D = 1$ and $C = 2$. This will make exercises with $D \leq 2$ and $C \leq 3$ available to the learner.

From here the student can access the solution interface for the selected exercise, and submit a program proposal in two ways.

The first way is to use a very plain interface, allowing to just paste the program and sending it along. At this stage only compiler-related feedbacks are available (if the program compiles well, it is send to the selected robot for execution). There is no simulation-baed check at this stage, to prevent patently wrong programs to execute, nor recovery programs are set into the system to let the robot return to a safe position afetr an unfortunate experiment. This is an operative lack in the system so far.

The second way to submit a solution is through a pseudo-code subsystem, that allow the student to develop the program in a guided manner, by specifying the sequence of movements and actions through menus and parameter sub-panels (Figure4).

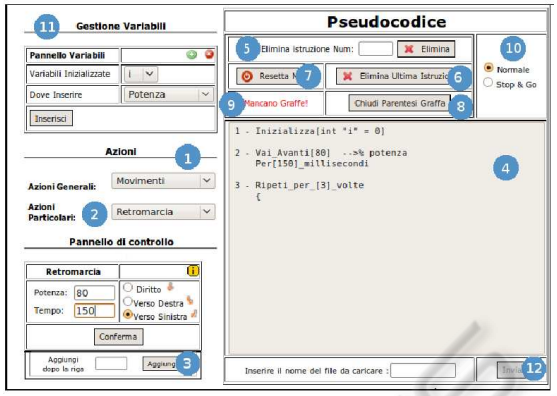

Figure 4: The pseudo-code interface. Menus allows to select the operations to list in the program. For certain operation a sub-panel opens up, to allow the setting of due parameters.

4 CONCLUSIONS

The paper has shown design and implementation aspects of the system MindLab, a web application supporting remote operation of LEGO Mindstorms robots for e-teaching and e-learning activities. The choice of such commercial hardware makes, without prejudice for the flexibility and the learning effectiveness, pretty affordable the task of making a laboratory available to distance learning, also in cases when the economic resources of a school are limited.

Our approach imports, in such a didactic teleoperation setting, our experience in the field of personalized and adaptive e-learning. In each course the teacher can define an exercising-learning path of mandatory exercises; the learner, then, can approach only those exercises whose pedagogical specifications are sufficiently close to the present cognitive state of the learner.

We have several robots available to remote exercising at the same time, both in separated environments (where different students can can operate with different robots in different portions of the laboratory, and for collaborative exercises with coordinated machines.

We have remarked the flexibility of the (small) exercise environment, with the possibility to define various layouts for different tasks, and the system capability to manage different robots, each one with its configuration, for different exercises. And we have pointed out some lacks in the present implementation of the system, namely the absence of any simulation devices (that could avoid execution of wrong plans)

and the scarce recovery mechanisms to enact in case of execution of such wrong programs, or other accidents.

In terms of future work we plan to remedy to the abovementioned lack, by equipping the system with dedicated modules. There are some other lines development: one is the definition of a more accurate, yet not more cumbersome, method to define the pedagogical characterization of an exercise; a second line of research is the definition of a more permanent student model, such as it is not entirely initialized when a new course is endeavored by the learner.

From an adaptive-e-learning point of view, then, we will try to formalize and implement a framework in which the sequence of exercises for a student, is configured according to the learner's individual cognitive state and learning goals, and possibly changed (i.e. *adapted*) during the course activity, responding to changes in the student model.

REFERENCES

Traylor R. L., Heer D. and Fiez T. S. (2003). Using an Integrated Platform for Learning to Reinvent Engineering Education. *IEEE Transactions on Educations*. Vol. 46, No. 4, pp. 409–419.

Gomes L. and Bogosyan S. (2009). Current Trends in Remote Laboratories. *IEEE Transactions on Industrial Electronics*. Vol. 56, No. 12, pp. 4744–4756.

López D., Cedazo R., Sánchez F.M. and Sebastian J.M. (2009). Ciclope Robot: Web–Based System to Remote Program an Embedded Real–Time System. *IEEE Transactions on Industrial Electronics*. Vol. 56, No. 12, pp. 4791–4797.

Balestrino A., Caiti A. and Crisostomi E. (2009). From Remote Experiments to Web–Based .Learning Objects: An Advanced Telelaboratory for Robotics and Control Systems. *IEEE Transactions on Industrial Electronics*. Vol. 56, No. 12, pp. 4817–4825.

García–Zubia J., Orduña P., López–de–Ipiña D. and Alves G.R. (2009). Addressing Software Impact in the Design of Remote Laboratories. *IEEE Transactions on Industrial Electronics*. Vol. 56, No. 12, pp. 4757–4767.

Kosuge K., Kikuchi J. and Takeo K. (2002). VISIT: A teleoperation system via the computer network. Beyond Webcams:an introduction to online robots. MIT Press, pp. 215–220.

Marín R., Sanz P. J., Nebot P. and Wirz R. (2005). A Multimodal Interface to Control a Robot Arm via the Web: A Case Study on Remote Programming. *IEEE Transactions on Industrial Electronics*. Vol. 52, pp. 1506–1520.

Colwell C., Scanlon E. and Cooper M. (2002). Using remote laboratories to extend access to science and engineering. *Computers & Education*. Vol. 38, pp. 65–76.

Lerman S. and del Alamo J. (2000-2005) iLab: Remote Online Laboratories. [Online]. available: http://icampus.mit.edu/projects/iLabs.shtml.

Chellali R., Dumas C., Mollet N. and Subileau G. (2009). SyTroN: A Virtual Classroomfor Collaborative and Distant E-Learning Systemby Teleoperating Real Devices. *International Journal of Computer Games Technology*. Vol. 2009, doi:10.1155/2009/627109. Hindawi.

Borgolte U. (2009). Interface design of a virtual laboratory for mobile robot programming. In *Mndez-Vilas, A., Solano Martn, A. , Mesa Gonzlez, J.A. and Mesa Gonzlez, J. (Eds.) Proceedings of the m-ICTE2009 Int. Conf. on Research, Reflections and Innovations in Integrating ICT in Education*. 22-24 Apr. 2009, Lisbon, Portugal. Vol. 1. ISBN 978-84-692-1789-4. FORMATEX, Badajoz, Spain.

Casini M., Prattichizzo D. and Vicino A. (2004). The Automatic Control Telelab. A Web-based technology for Distance Learning. *IEEE Control Systems Magazine*. Vol. 24, No. 3, pp. 36–44.

Poindexter S. E. and Heck B. S. (1999). Using the Web in your courses: What can you do? What should you do?. *IEEE Control Systems Magazine*. Vol. 19, No. 1, pp. 83–92.

Merrick C. M. and Ponton J. W. (1996). The ECOSSE control hypercourse. *Comput. Chem. Eng.*. Vol. 20, sup., No. 972, pp. S1353-S1358.

Schmid C. (1998). The virtual lab VCLAB for education on the Web. in *Proc. American Control Conference*. Philadelphia, PA, pp. 1314-1318.

Jara C. A., Candelas F. A. and Torres F. (2008). Virtual and Remote Laboratory for Robotics ELearning. in *Proceedings 18th European Symposium on Computer Aided Process Engineering ESCAPE 18, Braunschweig B. and Joulia X. (Eds)*. Elsevier.

Fernandez, G., Sterbini, A. and Temperini, M. (2007). Learning Objects: A Metadata Approach. In *Proc. IEEE Region 8 Eurocon 2007 Conference*. Warsaw, Poland.

Limongelli, C., Sciarrone, F., Temperini, M., and Vaste, G. (2009). Adaptive Learning with the LS-Plan System: A Field Evaluation. *IEEE Trans. Learning Technologies*. Vol.2, No. 3, pp.203–215.

Sterbini, A. and Temperini, M. (2009). Adaptive Construction and Delivery of Web-Based Learning Paths. In *Proc. 39th ASEE/IEEE Frontiers in Education Conference*. San Antonio, TX, USA.

Limongelli, C., Sciarrone, F., Temperini, M., and Vaste, G. (2010). *The Lecomps5 Framework for Personalized Web-Based Learning: a Teacher's Satisfaction Perspective*. Computers in Human Behaviour, Elsevier, to appear.

Bloom, B. S. (1964). *Taxonomy of Educational Objectives*. David McKay Company Inc.