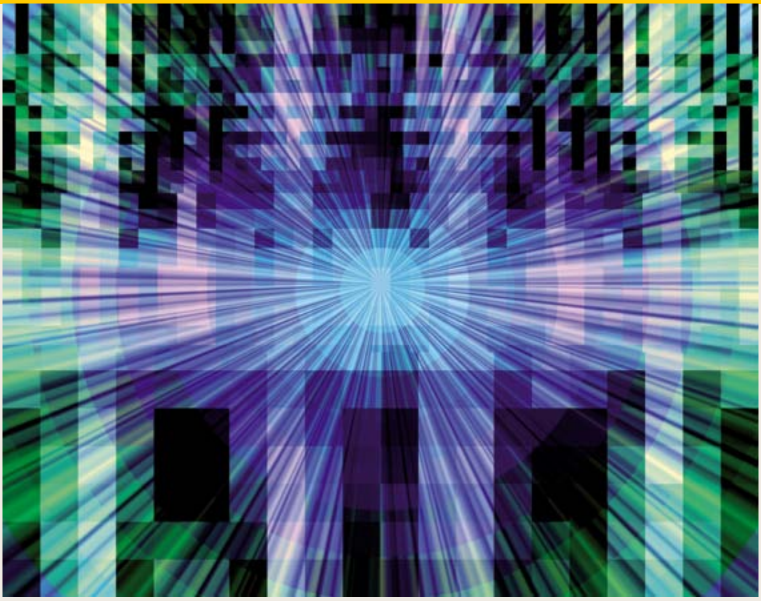
Olga Dziabenko and Javier García-Zubía (eds.)

IT Innovative Practices in Secondary Schools: Remote Experiments



Deusto

Remote laboratory for serious games deployment based on a mobile robot platform

Iñigo Iturrate, Ignacio Angulo, Pablo Orduña
Deusto Institute of Technology – DeustoTech, University of Deusto, Bilbao (Spain)
e-mail: ignacio.angulo@deusto.es

1. **Introduction and contextualization**

In the last few decades, technology has evolved at an ever-changing pace. With these developments, the need has also arisen for education to evolve and incorporate new contents and methodologies, but it has often found itself lagging behind. Recent studies and surveys have found that few students are interested in taking up careers in science, technology, engineering and mathematics (STEM), when compared to the current needs for professionals in these fields expressed by the industry [1]. As such, the educational community faces a pressing need to adapt to this reality and develop new curricula, platforms and resources to attract and inspire learners. It is the belief of some researchers intimately related to these fields that remote education and laboratories could present a solution to this problem [2].

Engineering education has seen Remote Laboratories grow rapidly in the past few years and becomes a true alternative to traditional experimentation and teaching methods. These remote environments seek to provide a viable substitute for or complement to on-campus laboratory exercises by connecting real physical experiments to the Internet, which means they can be accessed from anywhere in the world. In this way, students can continue working on their practical exercises from their homes or anywhere else with Internet access. Additionally, teachers do not see their practical lessons hindered by lab material unavailability or safety issues. One could ask oneself how this is different from a simulation. Here, the answer is that Remote Laboratories, contrary to virtualization and simulation, provide users with a real experiment. In theory, this means that a Remote Lab should be no different from physically measuring the indicators of an experiment –it will actually use a physical measurement device to obtain this reading, with the only difference being that this is then transmitted to a server that then sends this information back to the client, that is, the student. Moreover, remote laboratories pave the way for the "democratization" of education, whereby any person with access to the Internet can take part in the experiments. They cross national boundaries, economic boundaries, connect universities and reduce the time constraints of users. They allow students to continue their work from outside the university campus. Remote lab federation –the sharing of these environments across institutions– allows universities and schools to access equipment that could otherwise be well beyond their budgets or their space constraints. Another advantage of this approach to practical learning is that the more educational institutions there are that standardize them and replicate them in several locations, the more users that are simultaneously supported, and therefore the better the quality of the service provided.

Separate from the technical and engineering view of education is that of the pedagogical community. Here, a significant approach to renewing educational methodologies comes in the form of *Serious Games*. Supporters of this view aim to abandon lecture-based materials in favour of interactive content that also conveys learning in a context of entertainment and fun. There is long-standing consensus among pedagogues that, if appropriately applied, games can help to improve learning [3]. A key concept here is that of *flow*, defined as a state where consciousness of space and time are reduced, and the person becomes deeply focused on the subject at hand [4]. It is believed that by creating suitable game elements to surround education, the learner can enter such a state, which has been shown to enhance performance and also better engrain understanding of the concepts learned during the activity [5], [6]. In order to generate and maintain this state, however, it is necessary to carefully and properly design the learning game, so that it presents the equilibrium of several elements. Kiili, based on an Experiential Gaming Model [7], [8], identifies three such constituents of game design, namely: storytelling, game balance and optimization of cognitive load (Fig. 1). A story is fundamental to integrating gameplay within a larger context and can also be used to provide important factual information. In this case, it is important to properly connect the story to the actual gameplay, and not let it be a forced, artificial relation. Game balance is crucial for achieving player involvement and maintaining it throughout the game experience. In this sense, challenges should be tailored to the skill level of the player, which should be carefully monitored throughout the gaming session. Here, it is essential that the challenges require critical thinking and creative solution generation if the player is to learn properly; i.e. solutions should not be reached merely through memorization and repetition (unless this happens to be needed for the subject). Lastly, optimization of cognitive load has to do with the fact that our visual and auditory memory processing channels are limited [9]. In keeping with this, it is not advisable to overload the game player with too much visual and auditory information in the form of flashy graphics and sounds, since this can have a negative effect on their ability to learn. A balance between this must therefore be sought.

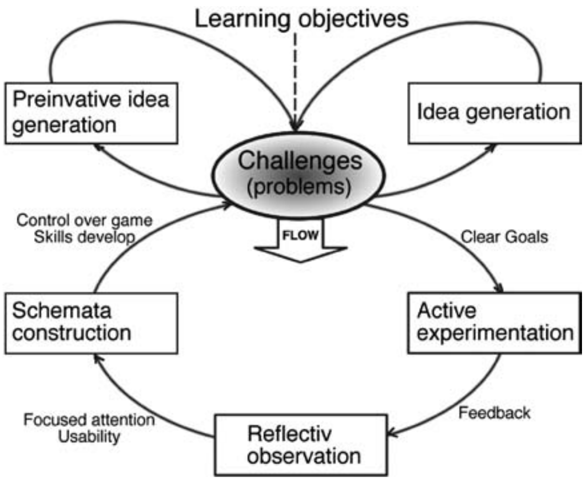


Figure 1

Kiili's Experiential Gaming Model

This chapter presents an initial approach to the blending of Remote Laboratories and *Serious Games* currently deployed at WebLab-Deusto at the University of Deusto, which attempts to apply pedagogical game design criteria to a remote experiment, with the objective of creating a platform for programming education. The experiment allows users to control a mobile robot in a labyrinth environment and take part in an interactive game where they must locate and correctly answer several questions, the subject of which can be adapted to educators' needs. It also integrates the Google *Blockly* graphical programming language, allowing students to learn basic programming and logic principles without needing to understand complex syntax.

2. **Previous work**

This project is a direct continuation of the subject of [10], which hypothesizes integrating remote laboratories and *Serious Games* by developing a platform where students must use a mobile robot to locate several items scattered around a maze. The story setting for the game revolves around a situation in the distant future in which a spaceship has been forced to make an emergency landing on an unknown planet. Seeing that the ship has lost several key components, which have been scattered on the planet's surface, and that these pieces are critical to rebuilding the ship and leaving the planet, the crew must deploy a reconnaissance robot to recover them. Once all the pieces have been retrieved, the spaceship can be repaired and it can leave the planet. However, there is also a need for fuel for the robot. In order to gain the necessary currency to buy this, the crew can answer questions as part of an intergalactic trivia contest. Furthermore, in order to succeed in the game, they must first learn to control and program the robot to navigate a hostile environment and find the pieces. The condition to end the game successfully is that of finding all (five) of the RFID tags representing the parts of the spaceship. Once all have been gathered, the ship can be repaired and leave the planet for good.

Currently, the implementation of the laboratory does not yet provide all of the game elements described above. Nevertheless, it sets the ground for them by providing a suitable environment for the development of the game. Moreover, it takes care of most of the technical requirements of the system. Thus, future work of the project can focus mostly on the pedagogical side, without the need for much technical development.

At its current state of development, the remote experiment allows manipulation of the robot in two different ways:

– Direct control of the robot can be exerted by means of arrows controlling movement in the primary directions. RFID tags representing different parts of the spaceship are scattered around the maze, and whenever the robot reaches one, it can be retrieved, and the user can answer a question related to various educational and cultural subjects.
– The robot can be programmed to navigate the labyrinth and search for the pieces autonomously. The player can use a graphical programming language (Google Blockly) to do so in an intuitive and unobtrusive way, while learning basic programming principles without the need to understand the syntax of a textual language.

3. **Experiment architecture and technologies**

There are three main layers that make up the architecture of the experiment and its technical implementation, each directly related to one of the main components of the WebLab-Deusto architecture:

– Microbot: This is the physical part of the experiment and the main component in the system. It consists of a small robot enclosed in a wooden labyrinth. The robot has the necessary sensors to follow lines, detect walls, and read RFID tags. The robot delegates most of its intelligence to the Experiment Microserver Control Computer, thus reducing its own duties to merely following orders received from the computer.
– Experiment Microserver: This computer is in charge of direct communication with and control of the robot. Orders are sent through Bluetooth indicating movements. The microserver also provides the user client GUI and controls. The Microserver is where the main intelligence of the robot lies: Google Blockly programs actually run locally on this machine, and are interpreted as a series of simple Bluetooth movement commands sent over to the Microrobot for execution.
– WebLab-Deusto: The WebLab-Deusto Server takes care of user authentication, queuing, security, system logging and several other

administration tasks, and directly communicates with the experiment microserver. It is based on the open-source architecture developed at the University of Deusto.

A. *Microrobot*

A key factor in design choices for the micro-robot was making the platform as low-cost as possible and easing maintenance and handling, in order to reduce possible experiment down-time due to repairs or the need to redesign part of the software. For this reason, the choice of parts was based on components provided by generally well-known manufacturers. Furthermore, Arduino was chosen as the main microcontroller system, due to its widespread popularity, open-hardware philosophy and ease and quickness of development.

The whole robot structure was based on the *MSE ArduBot* [11] kit, which uses the *Pololu RRC04A* circular chassis as the main building component. The *RRC04A* allows easy assembly of motors, sensors and other components, and is also designed to be able to be stack various copies of itself on one another, thus providing a modular building block that can be used to create various levels for component placement. The robot built in this project used three such chassis. The lower level housed the *Pololu 150:1 Micro Metal Gearmotors* included with the kit, as well as all the infrared sensors used for line detection (and navigation in the labyrinth) and the RFID reader module. The middle level included the *Arduino UNO* microcontroller, as well as a custom-designed *Arduino* shield including a motor driver, Bluetooth module and voltage regulators, which will be explained in the following sections. The top level was where the battery providing power to the whole robot and the webcam that transmitted the robot's view to the Experiment Computer was placed. A critical aspect in the location of the battery was balancing the robot. It was centred horizontally, but moved towards the back of the chassis so that it would compensate for the weight of the camera and to prevent the robot from tripping over forwards. However, in the end, the battery was found to be so heavy that, instead, the robot tended to tilt backwards; it was therefore necessary to add two follower drive-less wheels at the front and back of the robot's lower level to keep it steadily balanced on the ground.

Placement of the infrared sensors for line detection was a crucial feature of the implementation phase. Throughout development, several different sensor configurations were tried before the final one was chosen. Factors influencing the position of sensors are:

– Space constraints due to chassis design.
– The ability to precisely follow a black line.
– Detecting intersections and stopping the robot exactly in the middle of one of them.
– The speed at which the robot will move.
– Whether the robot is to move only forward, or also backward.

The final position of the infrared sensors can be seen in Fig. 2.

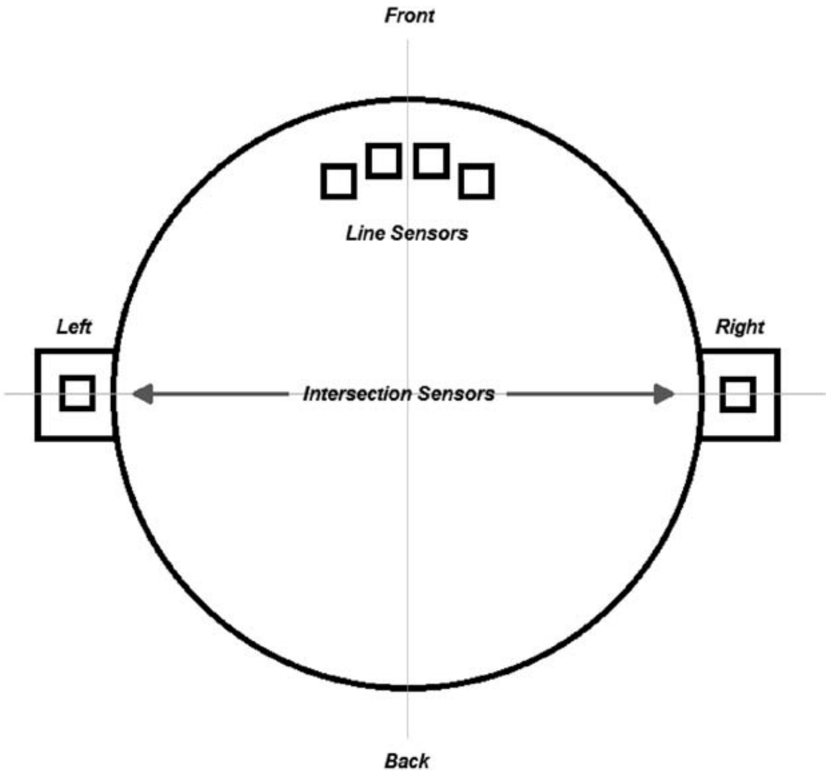


Figure 2

Sensor placement for the microrobot

All external electronics required by the robot were placed on a separate printed circuit board. To enable easy connection of the board with the Arduino UNO microcontroller, the PCB was fitted into one of the standard *Arduino Shield* form-factors, pin-by-pin compatible with standard Arduino microcontrollers. The shield was designed to house the electronics responsible for motor driving and control, Bluetooth communication and general voltage regulation and supply to all other sensors and electronic components. The chosen motor driver was the *Microchip MCP14E5*, based on MOSFET technology, thus providing very fast switch and response times. The circuit was designed to provide control lines for motor turn speed and direction and to incorporate an electronic brake.

The Arduino UNO board was programmed with a fixed software (firmware) incorporating the basic execution routines required by the robot. This basically consists of the robot following the line until it reaches an intersection, stopping at the intersection, attempting to read an RFID tag and sending it to the Control Computer, awaiting Bluetooth commands from the Control Computer, and then performing the actions indicated by these commands. Therefore, the robot's intelligence is delegated to the Control Computer and the robot simply follows the commands it receives.

B. *Experiment microserver*

The Control Computer, or Experiment Microserver, was programmed using Python. The philosophy behind this language makes it well-suited to quick development phases and also offers a very clear and readable syntax. The resulting code is usually simple and efficient, thus being optimal for ongoing development and for modular projects. There also exist a wide variety of useful software libraries that provide ready-built frameworks and components, facilitating many areas of the development process.

The core of the Control Computer is based on the Django Web Framework for Python. This eases the creation of web-services and the integration of databases within them. The software also uses the Pybluez library, which provides access to the Bluetooth stack in Linux systems under Python, and Twitter Bootstrap is used as a framework to provide HTML, CSS and JavaScript templates to ease in the implementation of the client, which was written as an HTML webpage.

Within Django, the *views.py* module is perhaps the principal component of the system. It is responsible for rendering the HTML pages, handling HTML requests and receiving and processing POST and GET commands. It also manages Bluetooth communications with the robot and receives commands from an optional *joystick_handler.py* module that allows the robot to be controlled with a joystick, but only if the lab is running locally. Finally, it implements the Google Blockly functionality that translates a series of stacked graphical program blocks into actual Python code. This code is then executed in the Control Computer and sends commands to the robot, processes information coming from its sensors and essentially allows the user to feel as if he were programming the robot with a series of graphical blocks. In reality, the program is not running on the robot's Arduino UNO, but on the Control Computer, corresponding to the delegation of intelligence described earlier. This greatly simplifies the implementation by removing the need to access the Arduino Bootloader every time the robot is re-programmed.

The remaining module, *urls.py*, is in charge of redirecting calls to the various sub-files within the server to the correct URL.

There are two main HTML files in the client: *index.html* and *blockly.html*. The first corresponds to the first instance of the game where the user directly controls the robots movement, while the second integrates the Google Blockly graphical programming workspace into the User Interface, corresponding to a more advanced player's skill profile (Fig. 3). The GUIs for these two modes of control will be described in following sections.

Google *Blockly*, included within *blockly.html*, is a graphical programming language very much in the vein of *Scratch* or *SNAP!*. It is visually very similar to *Scratch*, but differs significantly in its implementation: whereas *Scratch* is written in Squeak, a dialect of Smalltalk, and therefore requires the execution of a Smalltalk virtual machine in the host machine, *Blockly* is written entirely in JavaScript and is thus better adapted to the web. From the developer's point of view, *Blockly* allows easy creation of new graphical function blocks by writing them in straightforward JavaScript syntax (or using an online editor called *Block Factory*, written in *Blockly* itself), and then defining a series of Generators that translate these blocks into other text-based programming language. *Blockly* includes methods that generate code from the blocks and translate it into JavaScript or Python, which can then be executed as a normal program file (Fig. 4).

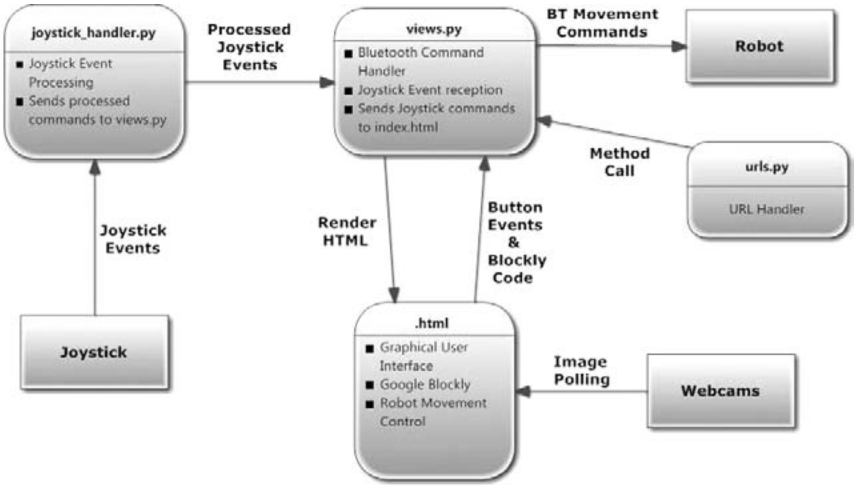


Figure 3

Architecture of the experiment.

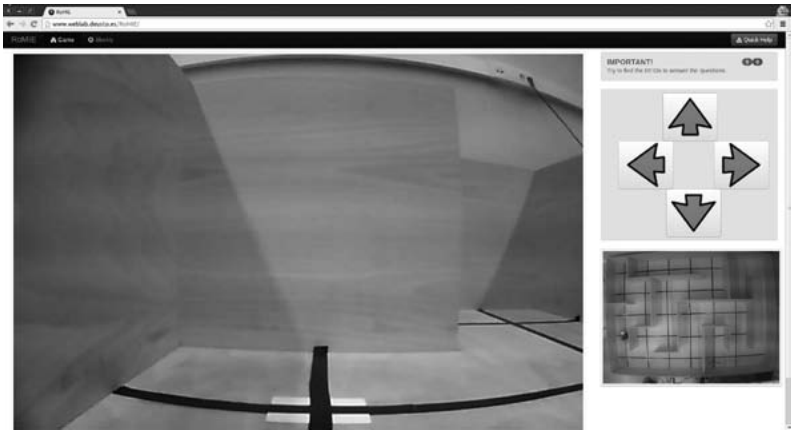


Figure 4

Movement Control GUI for the experiment

Within *blockly.html* is the Google Blockly client (Fig. 5). This is a graphical programming language that allows users to create fully functional programs by dragging and joining coloured blocks that carry out specific functions. There are blocks for logic statements, for loops, for conditions, for arithmetic, etc. This is similar to other graphical programming environments, such as SNAP! or Scratch. The latter were also considered for the project, but they were discarded as they incorporate too many restraints on the developer. Some of these were for security reasons, but a significant fact was also that they are written in Squeak, a dialect of Smalltalk, which requires the execution of a virtual machine in order to deploy the client, thus greatly complicating development. Blockly, on the other hand, is written entirely in JavaScript, making it optimal for use in web applications. This language also makes it incredibly simple to develop new graphical blocks by either writing them in JavaScript or using an editor called Block Factory (which incidentally uses the Blockly GUI itself). A set of blocks can then be easily translated into JavaScript, Python or XML by defining a set of language generators. This generated code can then be executed on the Experiment Microserver, which sends a series of commands to the robot.

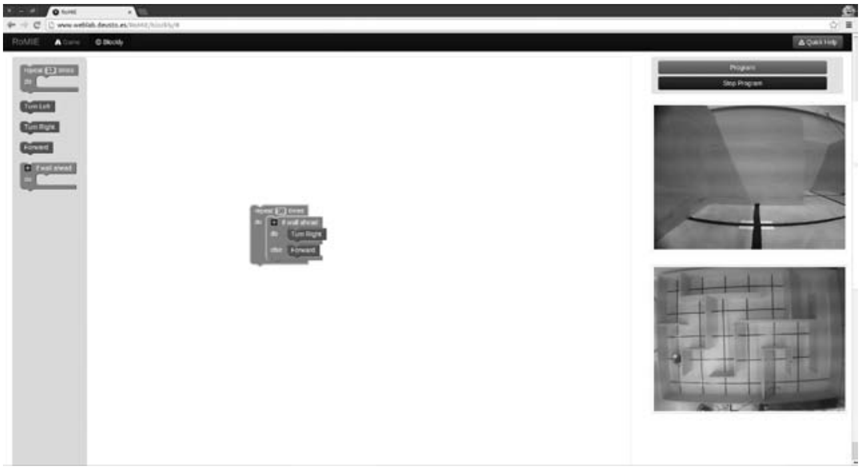


Figure 5

Graphical User Interface integrating Google Blockly with the robot

An important consideration when accepting code from third parties is safety issues and preventing the injection of malicious code. For this purpose, the implementation of Blockly deployed at WebLab-Deusto has several security checks when translating block code. First, the blocks are translated into XML. The very explicit syntax of this language makes it easy to check for dangerous segments, and to filter the code so that it only accepts a series of very well defined statements. Once the XML has been properly checked and filtered, it is fed into a Python interpreter, which translates it into a structure of function calls related to robot functionality, e.g. move forward, turn left, turn right or check a sensor. This Python code is then written to a temporary file, which is then executed as a separate process within the Experiment Microserver. In order to prevent infinite loops and lock-up of the machine, the computer also incorporates a timeout, which kills the process if it exceeds a pre-specified amount of time. Additionally, the user may also choose to terminate the program at any time by clicking on the "Stop Program" button.

C. *WebLab-Deusto*

The WebLab-Deusto framework [12] is the basis for this remote experiment. The WebLab-Deusto server is connected to the Robot Control Computer and allows easy implementation of laboratory management functionality simply by using the WebLab API. In essence, if an experiment is deployed as a managed laboratory under WebLab-Deusto, it will benefit from the following advantages [13]:

– There is no need for the experiment developer to manage communications, as these are taken care of by methods in the client API. These propagate information to the server securely under SSL. All commands are pure HTTP, so they can cross any firewalls and proxies.
– Storage of used information in a database. This is useful for teachers and pedagogues to track use of the experiments and commands sent to the laboratory by their students.
– Local load balancing, so that, in the case that there exist multiple copies of the same laboratory, the WebLab-Deusto server will internally manage messages sent by the client to the appropriate copy of the laboratory that is currently being used.
– Authentication of users is automatically handled by the WebLab-Deusto server.
– Queuing is internally handled by WebLab-Deusto.

– Sharing of laboratories among different institutions *–Remote Lab Federation–* is supported, and is a major design goal.

4. Educational Value

The developed experiment provides an educational tool now applicable in three different scenarios:

1. Development of multidisciplinary questionnaires.
2. Experimenting with programming skills using a block-based graphical interface.
3. Experimentation with control structures of different high-level programming languages.

For the first scenario of the game a plot is proposed:

On the way to Triangulum Galaxy, the control devices of an inter-galaxy ship have begun to work unpredictably and strangely. The first pilot of the expedition decided to stop on the closest planet so the team could rest, research the planet environment and repair the equipment. During the landing, the environmental robot Max375v.11075 (or simply Max) was destroyed and his parts were scattered across the surface of the planet. From the first environment research, the expedition team understood that the air on this planet was highly acidic and the robot, Mini200v.468 (or simply Mini), should find all parts necessary to reduce the damage to Max in a very short time.

The task of student is to find, collect and assemble all the parts of Max in the correct way. The game is multidisciplinary, supporting different subjects for teaching and learning. For instance, the plot presented above is applicable to the teaching of Technology in secondary schools. Through a game authorizing tool a teacher can modify the task that student should collect and put together. A Chemistry assignment could be used to find necessary parts of chemical elements and compound them into the correct chemical formula. The English teacher can create vocabulary allowing students to collect only those words that could be used for building meaningful sentences or even a brief story. Using the graphical interface enables direct control of the robot, and any teacher can design a support questionnaire relating to their subject by defining a set of questions and their answers. These questions can be grouped into different categories, each linked to one RFID tag. Every time the robot detects a tag (in the game it is part of Max), a question related to the corresponding category will pop up. The teacher can set the initial number of robot movements each student is allowed, game-time restriction and the reward the player gets every time he answers a question correctly. The student has to complete the game answering at least one question of each category correctly, collect the parts of Max and assemble him (Fig. 6). Those users with a teacher role can access a control panel where they can define the questionnaire and review the results of all experiments performed by the students in the same group. In this approach the serious game is only the vehicle that provides the questionnaire and controls the times and movement of robot *Mini*. GUI Movement control allows the student to find different tags and, every time a question is correctly answered, a new part of the spaceship is recovered. Once all missing parts have been collected (the entire questionnaire is filled in) or the appointed game-time is up, the game is over and the teacher can access all information about the robot movements, number of errors when answering questions and time used by each student to complete the game. As users are grouped, different teachers can provide independent questionnaires to students from different classes but access the same remote laboratory (robot *Mini)*.

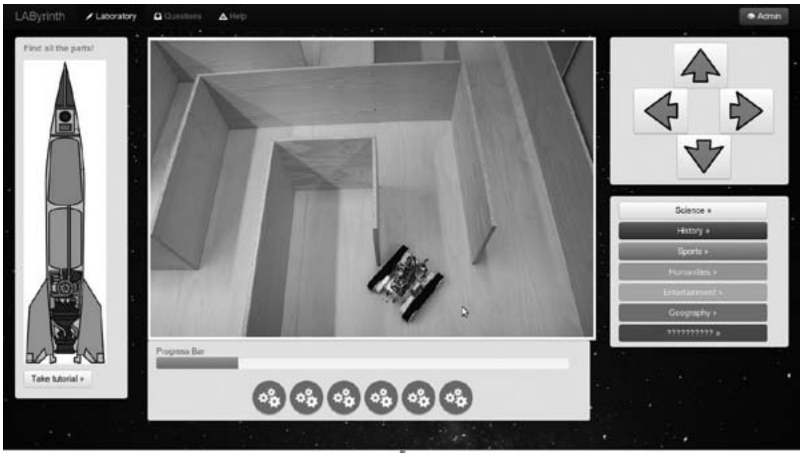


Figure 6

Questionnaire GUI

The second scenario is based on integrating programming principles such as logic and code writing into the teaching concept with a simple game setting. The environment suitable for the learning of basic programming skills and for introducing young students to the logic required from an engineering professional is developed. At the same time, the acquisition of programming abilities is intended to be gradual and objective-driven, with students seeing the codes as a means of reaching the goals of the game, thus presenting the programming discipline as a means of problem-solving and as eminently practical. Furthermore, by providing a responsive and real environment in the form of a mobile robot, users receive immediate feedback that they can use to judge their performance within the game. Since the robot operates in a real environment, the feedback that the user receives is consistent with this scenario (it is also real), and this means that the user's debugging experience and dealing with his mistakes is deeply enriching and promotes learning from and working on improving the mistakes made. This is also intended to act as a reward system of sorts when students can see that they are able to successfully tell the robot how to autonomously navigate the areas of the labyrinth.

The choice of a graphical programming language over a textual one allows a larger audience to use this activity, since the limitation of the programming-language-literate is reduced with the Google *Blockly* approach. This means that people of younger ages, as well as enthusiasts that lack engineering backgrounds, can also benefit from the game-laboratory. Furthermore, the complications of syntax are eliminated, bringing logic and not language to the spotlight. It is this logic, and not the strictness of syntax, that is the key to developing the right mindset needs of engineering professionals.

Another advantage of the Google Blockly graphical programming language implemented in the remote experiment is that it can easily be tailored to suit the needs of a particular laboratory session or user. Several difficulty levels or practice sessions can be implemented where the user is limited to a particular number or type of function blocks. For instance, a level can centre on the use of conditional statements, branched conditionals; another on conditional loops, another for loops executed a specific number of times, for loops with changing starting and ending conditions, etc. In this way, logic concepts and the programming of students' vocabulary can be introduced gradually and in an incremental fashion, providing a sense of empowerment by actually making complex things that work with simple elements, and by working up step by step to something powerful and more advanced.

At the same time, the addition of a real robot to this platform is a significant element when it comes to student engagement. The robot provides a real goal-driven testing ground for the programs. Having assembled the blocks, users can execute the code on the robot, thus seeing the response of a real system to their commands. By providing a back story and a clear objective to be achieved (in the form of finding the five RFID tags –spaceship parts– needed to rebuild the spaceship), the experiment aims to motivate and reward clear critical and logical thinking. The fact that there are varying levels of complexity for the blocks involved and their function makes it suitable for a wide variety of players from different skill levels and backgrounds.

5. **Conclusion**

If the STEM educational community is to appropriately react to the needs of the industry, it must find new ways to attract students and new methods to teach them the concepts that will be required of them as professionals. The merging of remote laboratories and *Serious Games* could perhaps be one such way.

WebLab-Deusto has developed a platform integrating game elements within the context of a platform for the teaching of basic programming logic within a STEM education context, without the need for students to first learn tedious programming syntax. As a remote laboratory, it allows ubiquitous access through the Internet, and is connected to real equipment that responds in a real way. Like the foundations of a *Serious Game*, it provides a goal-based environment for the development of learning objectives. This experiment's current state of development allows users to use the Google Blockly graphical programming environment to combine function blocks that translate into code what allows a mobile robot to autonomously navigate a maze, while finding and reading a series of RFID tags. This is set in the context of a game: a spaceship has crashed on a lost planet and must recover all five of its scattered parts (tags) to complete the game. In order to do this, students first need to learn the logic structures used in programming.

Even if the technical part of the experiment is mostly complete, there is still much future work left to do in the project. This will mostly centre on improving *Serious Games* design methodology and applying it to remote experimentation, with the objective of fully immersing students in an entertaining experience that enhances their critical thinking programming capabilities and deeply focuses their attention. When the main gaming elements have been incorporated into the laboratory, it is the objective of the project to deploy it and test it in several schools in order to verify and further improve its usability and teaching effectiveness.

The possibilities of merging *Serious Games* and Remote Laboratories, which have been clearly distinct lines of research up to now, show great potential for the STEM educational community, and could significantly redefine the methods currently employed and open new doors to study methodology.

References

[1] Report. *Economics and Statistics Administration, United States Department of Commerce*. Commerce.gov, Sept. 2011., Web. 21 Sep 2011. <http://www.esa.gov/Reports>.

[2] O. Dziabenko, and J. García-Zubia. (2011) "Remote Experiments and Online Games: how to merge them?" In *Global Engineering Education Conference (EDUCON), 2011 IEEE*, pp. 1102-1107. IEEE.

[3] Norman, Donald A. (1993) *Things that make us smart: Defending human attributes in the age of the machine*. Basic Books (AZ).

[4] M. Csikszentmihalyi, and I. S. Csikszentmihalyi, eds. (1992) *Optimal experience: Psychological studies of flow in consciousness*. Cambridge University Press.

[5] M. Csikszentmihalyi, I. Csikszentmihalyi. (1975) *Beyond boredom and anxiety: The experience of play in work and games*. San Francisco: Jossey-Bass.

[6] Webster, Jane, Linda Klebe Trevino, and Lisa Ryan. (1994) "The dimensionality and correlates of flow in human-computer interactions". *Computers in human behavior* 9, no. 4: 411-426.

[7] K. Kiili. (2005) "Digital game-based learning: Towards an experiential gaming model". *The Internet and higher education* 8.no. 1 pp:13-24.

[8] K. Kiili. (2005) "Educational game design: Experiential gaming model revised". *Tampere University of Technology, Research report* 3.

[9] R.E. Mayer,, and R. Moreno. (2002) “Aids to computer-based multimedia learning”. *Learning and instruction* 12, no. 1, pp:107-119.

[10] O. Dziabenko, J. García-Zubia, and I. Angulo. (2012) “Time to play with a microcontroller managed mobile bot”. In *Global Engineering Education Conference (EDUCON),* pp. 1-5.

[11] Ingeniería de Microsistemas Programados, S.L., http://www.msebilbao.com/tienda/product_info.php?products_id=720, (accessed June 14, 2013)

[12] J. García Zubia, P. Orduña, D. López de Ipiña, G. Alves. (2009) “Addressing Software Impact in the Design of Remote Labs”. *IEEE Transactions on Industrial Electronics*. ISSN: 0278-0046; vol. 56, Issue 12, pp. 4757-4767.

[13] P. Orduña, J. García-Zubia, J. Irurzun, D. López-de-Ipiña, L. Rodriguez-Gil. (2011) ”Enabling mobile access to Remote Laboratories”. *IEEE EDUCON 2011*.

Olga Dziabenko and Javier García-Zubía (eds.)

IT Innovative Practices in Secondary Schools: Remote Experiments

Technologies play key roles in transforming classrooms into flexible and open learning spaces that tap into vast educational databases, personalize learning, unlock access to virtual and online communities, and eliminate the boundaries between formal and non-formal education. Online –virtual and remote– laboratories reflect the current IT trend in STEM school sector. The book addresses this topic by introducing several remote experiments practices for engaging and inspiring K12 students.

Lifelong Learning Programme

Deusto
University of Deusto