Turtle Sensors

How open hardware and software can empower students and communities

Tony Forster, Guzmán Trinidad, Andrés Aguirre, Facundo Benavides, Federico Andrade, Alan Aguiar, Gonzalo Tejera, Walter Bender

Abstract

We described an open approach to the use of the laptops for exploring science and engineering that is not predetermined or prescribed. Rather, their use is intended to be emergent, where change is a byproduct of teachers, children, and their parents learning together, discovering new possibilities, and sharing those discoveries. These open designs empower both the students and the communities in which they live. Communities and students in developing countries can be independent of imported equipment and technology for their science experiments. Students and communities are given back control over their learning. We postulate that students who are in control of their learning are more effective learners.

1. Introduction

The XO laptop[1] is designed with an emphasis on openness and empowering the end user. The laptop was designed to facilitate simple user-crafted extensions, such as low-cost sensors that plug into the external microphone-in jack or low-cost I/O boards. This allows a number of science experiments to be done with negligible cost beyond the laptop itself: the laptop can operate as a data logger through a range of sensors which can be built with village technology at negligible cost; and its ability to control actuators and read sensors can be enhanced if coupled with a low-cost I/O board, such as Arduino[2].

The Sugar Learning Platform[3], the software that runs on the XO laptop, also puts an emphasis on openness. It is Free Software licensed under the General Public License (GPL)[4] that permits unrestricted modification by the end user. Sugar goes a step further than most free software in this regard in that it provides built-in affordances for end-user duplication and modification. Users don't just have the license to make changes, they are encouraged to make changes.

These open designs empower both the students and the communities in which they live. Communities and students in developing countries can be independent of imported equipment and technology for their science experiments. Students and communities are given back control over their learning. We postulate that students who are in control of their learning are more effective learners.

There is a long tradition that supports this postulate, including the MIT 6.270 class, developed by Randy Sargent and Fred Martin, where students build and programmed robots[5], a format that has been widely replicated, for example, in the First robotics contests[6]. Other examples include the MIT 8.01X first-year physics course, which introduces students to classical mechanics, “uses a hands-on focus, and approaches mechanics through take-home experiments” (the students build their own apparatus).[7] The students often skipped lectures, but did their experiments.

A further benefit of open projects is that they can be decentralized and multinational. This paper has been written by authors in the United States, Uruguay, and Australia. It describes their independent but cooperative educational projects.

2. Learning through doing

In the conventional classroom, the teacher possesses knowledge “objects” (for example, facts about numbers and addition—the nouns and verbs of mathematics) which are handed out to the student. To be educated is to accumulate these objects, i.e., education is something that is done *to* the student. However, there is an alternative to passively receiving a body of knowledge: we focus on helping children to appropriate knowledge by putting it to work on problems that are meaningful to them. As a consequence, learning is something done *by* the student. In this type of learning, a child will not simply know the name of a thing and how it might be used in a narrowly defined context; they will understand its utility and its limitations. Put more succinctly, “You learn things through doing, so if you want more learning, you want more doing.” This is not a new idea: much of the groundwork was laid by John Dewey more than a century ago in his work on experiential education and hands on learning that emphasized active participation by the learner.[8]

A more contemporary vision of the potential of learning through doing can be found in the papers Seymour Papert wrote in the early 1970s on teaching children mathematics. Teaching math to young children in the abstract, divorced from other activities that are meaningful to them, he argued, was not effective. Indeed, Papert was not looking for improvements to long-tried methods of teaching algebra and geometry. Instead, he wrote:

> I am asking whether one can identify and teach (or foster the growth of) something *other* than algebra or geometry, which, once learned, will make it easy to learn algebra or geometry. . . In our ideal of a school mathematical laboratory the computer is used as a means to control physical processes in order to achieve definite goals. . . for example as part of auto-pilot

> system to fly model airplanes, or as the 'nervous system' of a model animal with balancing reflexes, walking ability, simple visual ability and so on. To achieve these goals mathematical principles are needed; conversely in this context mathematical principles become sources of power, thereby acquiring meaning for large categories of students who fail to see any point or pleasure in bookish math.[9]

Papert's insight was that the feeling of power—which might also be described as autonomy or independence—experienced by students *using* math to achieve interesting or important goals provides a fast track around the resistance to learning math in the abstract. Power is a satisfying and motivating feeling that makes learning easy. Helping children to make an emotional connection to learning is a crucial pedagogical goal that the XO laptop and Sugar have attempted to facilitate.

2.1 Programming as a place for doing (and redoing)

Forty years of working with children and computing has demonstrated that programming presents an opportunity to engage in open-ended problem-solving—wielding power—in a safe environment: there is nothing to break other than your program, and when it breaks you have the opportunity to find and fix the problem. The process of repairing or "debugging" a program—which Cynthia Solomon described as the great educational opportunity of the 21st Century[10]—provides a basis for active learning through trial and error. Success at fixing a program also gives students confidence that they can apply the same skills—defining problems, developing hypotheses, creating tests and executing solutions—to other problems they may encounter. Since it was our intent to raise a generation of innovators, we advocated the use of programming as a safe place for children to gain an understanding of and tolerance for the value of learning from mistakes.

2.2 Beyond "black boxes"

OLPC and Sugar go a step further than other computing environments in that it makes the source code of *every* program *immediately* available to the user. The premise is that that by taking something apart and reassemble it in different ways is a key to understanding it. In Sugar there are no black boxes: the learner sees what something does and how it does it. With just one keystroke, the Sugar "view source" feature allows the user to look at any program they are running (the word processor, video capture, web browser, calculator, etc.) and also to modify it. We go beyond "Beyond Black Boxes"[11] in that we not only make Sugar visible, we provide multiple affordances to the user for making changes.

The ability to not only *learn with the machine and software but also to manipulate and change the software and hardware* itself opens the door to learning lessons far more important than those necessary to pass a test. It leads children to the discovery that they are authentic problem-solvers in the real world. When the president of Uruguay, José Mujica, learned that a twelve-year-old from a small town east of Montevideo had written a half dozen activities for the XO he smiled and said triumphantly: “Now we have hackers.” In his eyes, this child’s ability to contribute to the broader development community was a leading indicator of change and development in the country.

Rogoff[12] describes the progression of community learning through the planes of apprenticeship, guided participation, and participatory appropriation. We have been building technological affordances that foster the creative connections to learning and that help community members to progress through Rogoff’s planes.

3. About the laptop

The XO laptop (See Figure 1) is developed by OLPC (One Laptop Per Child). The software which runs on it is produced by Sugar Labs, a member project of the Software Freedom Conservancy.[13]

Figure 1: The XO laptop

The XO laptop has the following built-in sensors: camera, microphone, touch pad, keyboard, buttons and game pad, magnetic-field sensor, thermometer, WiFi radio receiver, light sensor, and accelerometer.[14]

The external microphone socket can also be used as a DC-coupled data-logger input. The sound card in the XO hardware was designed to allow software selectable AC or DC coupling. This seemingly simple decision (inspired by Papert) meant a significant opening of the possibilities of the XO to connect to the physical world through measuring DC voltage and resistance. This possibility exists only in the XO, as any other computer has the microphone input AC coupled. In Uruguay, we are promoting the design, construction, calibration and programming of these sensors to foster creativity and empowerment of children and young people with their XO (See Figure 2).

The laptop can connect to a low-cost I/O board through one of three available universal serial bus (USB) connectors to expand its ability to interface with sensors and actuators. USB has been used to connect an RFID reader, Arduino, Lego Mindstorms NXT[15], Lego WeDo[16], GoGoBoard[17], and other I/O devices (including a portable ultrasound).

3.1 Physical quantities that can be measured through the microphone input

The XO, through its external microphone input circuit, can measure DC voltages in the range of 0.4 to 1.85 volts (approximately) and resistance between 700 and 14000 ohms (See Figure 3). Readings can be taken with a maximum sampling rate of approximately 20 Hz. This allows the direct measurement of these physical quantities. The measurement capability of the computer can be further expanded through the use of transducers: electronic components that translate physical quantities such as light, temperature, magnetic-field strength, etc., into voltage or resistance. Choosing the type of transducer to use depends on their local availability in stores (or on the street) and the component price. Another consideration is safety. Unlike resistive transducers, which do not pose any risk, some voltage transducers pose the risk of delivering inappropriate voltages that could cause irreparable damage to the XO. In our classroom experience, we have put no restrictions on the use of resistive sensors, but provide careful supervision of voltage sensors. We only work with older children (teenagers) with the latter.

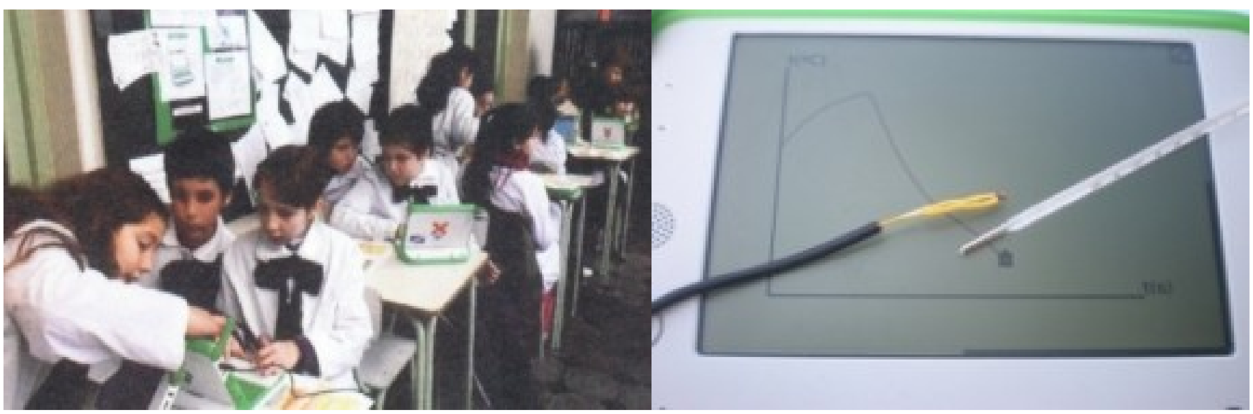

Figure 2: Students in Uruguay exploring sensor input (left); sensors and sensor output displayed on an XO (right)

Figure 3: The Measure activity is an oscilloscope that can be used to view analog audio and DC input.

3.2 Empowering the user with open technology

The software which runs on the XO laptop is Free Software: anybody is free to copy or modify the software. The hardware is less open but the philosophy of empowering users by giving open access to information still permeates OLPC hardware.

The programs which run on the XO laptop come with an explicit invitation to experiment. The programs (termed Activities) are written in Python[18]. Because Python is an interpreted language, the source code can be easily viewed and edited by users. The view-source mechanism described above allows source code to be viewed with a single mouse click. A second mouse click, from the view-source window, creates a duplicate of the code available for end-user modifications (See Figure 4). Because Sugar employs a “copy on write” approach to modifications, there is no danger that the end user break an installed program. Thus the penalty for “taking a risk” by modifying code is essentially nil.

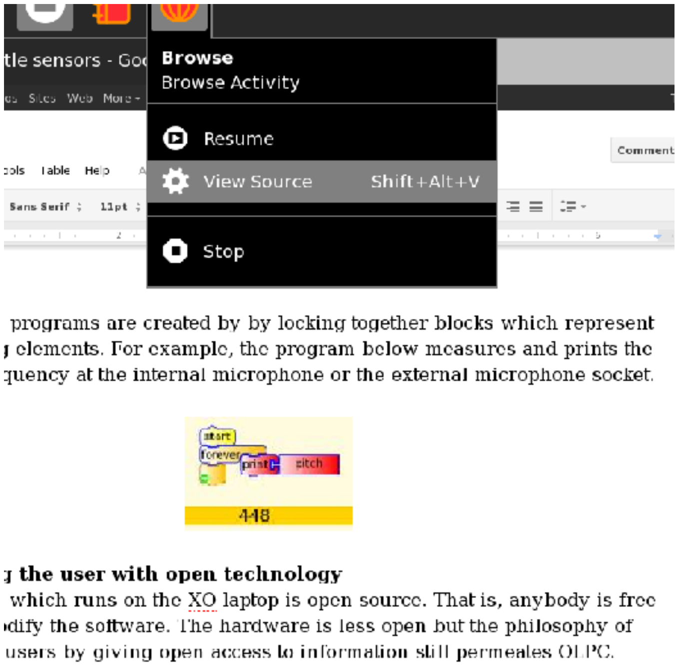

Figure 4: Viewing the browser source (while editing this document)

4. Turtle Blocks

Turtle Blocks[19] is a Sugar activity with a Logo-inspired[20, 21] graphical "turtle" that draws colorful art based on snap-together visual programming elements. Its "low floor" provides an easy entry point for beginners. It also has "high ceiling" programming features which will challenge the more adventurous student. As in most Logo environments, in Turtle Blocks, the turtle can exists in three forms: (1) as a robot sharing the same physical space as the child; (2) as a computational object that moves on the screen; and (3) as an abstract mathematical entity.

Turtle Blocks is a derivative of Brian Silverman's Turtle Art program[22], designed to further explore the richness of the Sugar environment and the XO laptop. Unlike Turtle Art, Turtle Blocks includes blocks for interfacing to sensors, access to multimedia, and additional programming features, such as a stack, nameable variables and procedures, and in-line access to Python. (The concept of in-line extensions has previously been explored in GameMaker[23], which provides an Internal code editor.)

Many of these features have been added at the request of and/or with the help of teachers. For example, a teacher in Uruguay requested the addition of a square-root block as she wanted to use it in teaching the Pythagorean theorem. At the time, adding a new block was tedious and beyond the abilities of most teachers, so it was added by the package maintainer. As will described in the next section, the current version of Turtle Blocks makes such requests trivial to fulfill. A great deal of effort has been made to extend an invitation to experiment with Turtle Blocks, and thus to become an active agent in one's learning.

4.1 Turtle Blocks with sensors

One of the easier way to access sensors is through the graphical programming language, Turtle Blocks (See Figure 5).

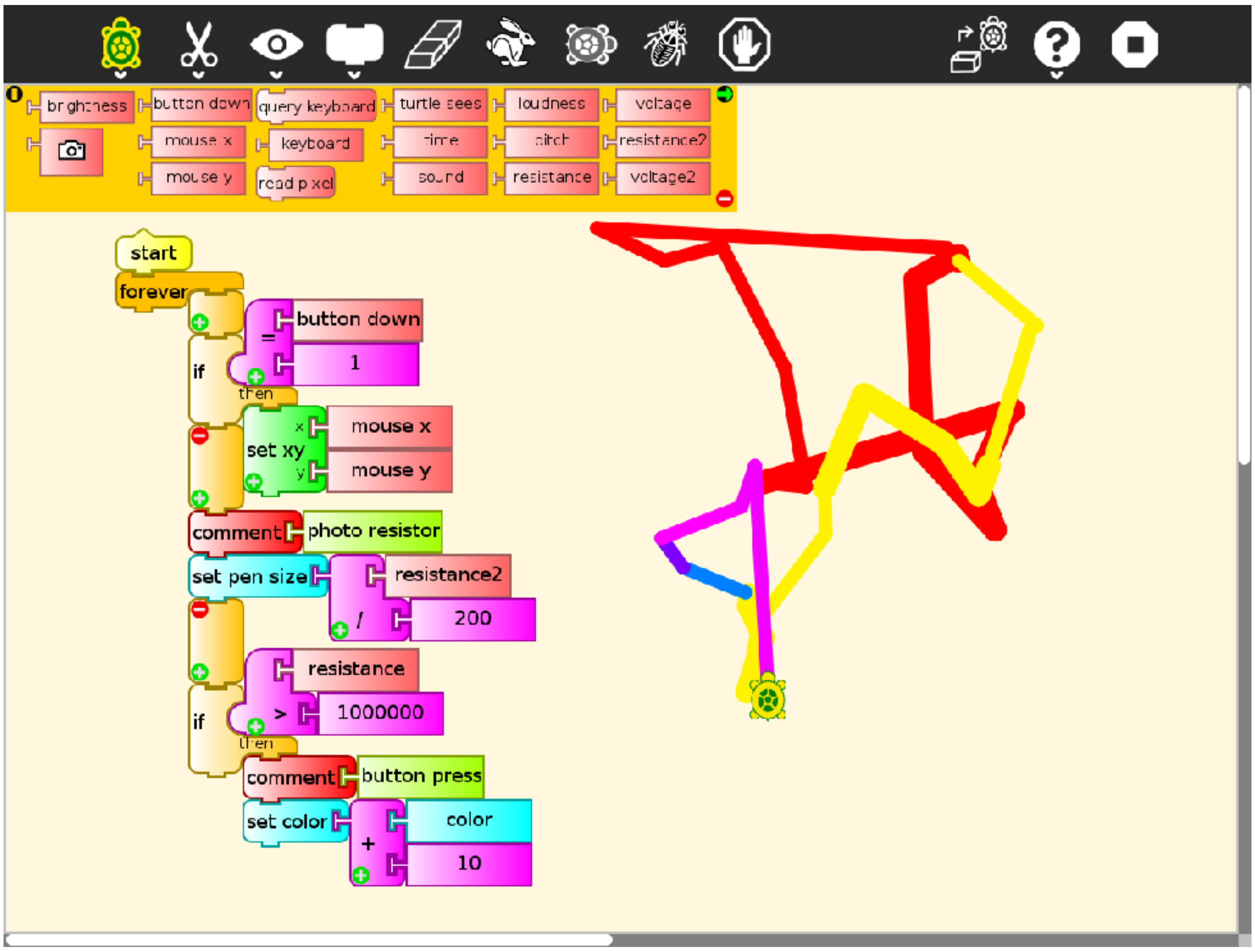

Figure 5: Accessing sensors with Turtle Blocks (in this example, sensors are used to change the pen size and pen color in a simple paint program).

Turtle Blocks programs are created by locking together blocks which represent programming elements. For example, the program shown in Figure 6 measures and prints the dominant frequency at the internal microphone or the external microphone socket. (The frequency of the strongest component is reported in Hz. The resolution is +-8Hz .) This simple stack of blocks has been used in a guitar-tuner program.

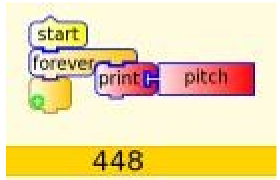

Figure 6: Measuring dominant frequency

4.2 Modifying Turtle Blocks

Turtle Blocks comes with built-in blocks for interfacing to sensors and it has two additional mechanisms for connecting to new devices:

- Python code block
- Plugins

4.2.1 Python code block

An obvious use for the DC-coupled microphone socket is to make a doorbell/burglar alarm, a switch can be constructed from two wires. This is a project with negligible material cost and suitable for village-level technology. The laptop senses the resistance at the microphone socket and sounds an alarm when the resistance falls below a predetermined level.

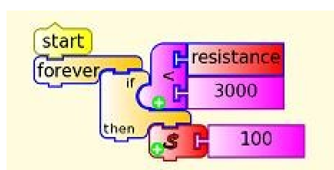

Figure 7: A simple door bell implemented in Turtle Blocks on the XO laptop

The way the alarm is sounded is an example of how an open-design philosophy empowers learners rather encouraging the belief that computer software is a ‘black box’ which is inaccessible to the user. At the time that this program was written, Turtle Blocks did not have a block to sound an audio tone, but because of the open design, one can be created. The following Python code in the Python Block sounds a tone:

```
def myblock(tw, x):
    import os
    os.system('speaker-test -t sine -l 1 -f %d' % *int(x))
```

Because the Python-encoded sound block was found to be so useful, it was eventually added to the standard Media Block Palette, an example of how end-user modifications can work their way “upstream” to the “master branch” of the project (See Figure 7).

4.2.2 Turtle Art plug-ins

Turtle Blocks provides support for plug-ins, “a set of software components that adds specific abilities to a larger software application... Plug-ins enable customizing the functionality of an application. Plug-ins are commonly used in web browsers to display new file types.”[24] The idea behind Turtle Block plug-ins is to let 3rd-party developers add new palettes and blocks to support additional functionality without having to make changes to any of the core Turtle Block packages. If a plug-in is present, it is loaded when Turtle Blocks is launched and any palettes or blocks defined by the plug-in are made available to the user.

The plug-in mechanism is currently used to provide support for sensors, the camera, RFID, and the Media, Extras, and Portfolio Palettes. A plug-in has been developed for Arduino, LEGO NXT, and GoGo. Another plug-in interfaces Turtle Blocks to a 2-D physics simulation engine, Box-2D[25].

4.3 How to write a plug-in

To add a plug-in, simply drop a file into the Turtle Blocks plug-ins directory.

Adding a new palette is simply a matter of defining a block and adding an associated method to call when the block is run. For example, to add a new turtle command, 'uturn', we use the add_block method in the Palette class.

```
palette.add_block('uturn',              # the name of your block
                  style='basic-style',  # the block style
                  label=_('u turn'),    # the block label
                  prim_name='uturn',    # the code to run
                  help_string=_('turns the turtle 180 degrees'))
```

And we define what the block will do, in this case, add 180 to the current heading:

```
# def_prim takes 3 arguments: the primitive name, the number of
# arguments, and the function to call.
self.def_prim('uturn', 0, lambda self: self.tw.canvas.seth(
                                   self.tw.canvas.heading + 180))
```

That's it. When you next run Turtle Blocks, you will have a u-turn block on the Turtle Palette (See Figure 8).

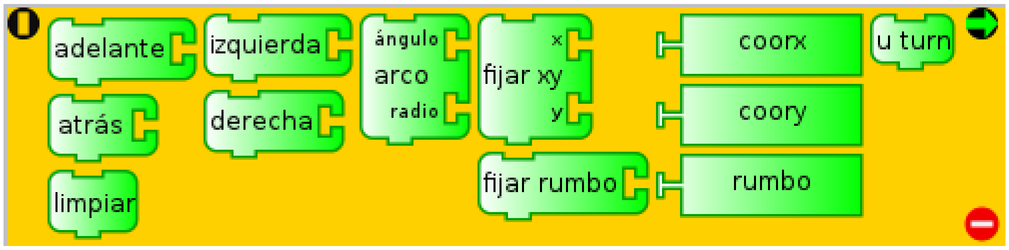

Figure 8: Adding a u-turn block to the Turtle Palette

Our emphasis on extending or reprogramming Turtle Art is pragmatic: the sensor and robotic technology available to teachers and children varies from school to school and from one OLPC deployment to another. To support the variations from a central point would require more time than the maintainers can commit. A decentralized model has proven itself again and again, as local groups extend support to meet their needs. We also have a pedagogical reason for this emphasis: we want to facilitate the progression of the community of learners to Rogoff's participatory appropriation stage.

5. Sensors created with village-level technology

Because the XO can log data through its DC-coupled microphone socket, low-cost sensors can be built (See Figures 9 and 10).

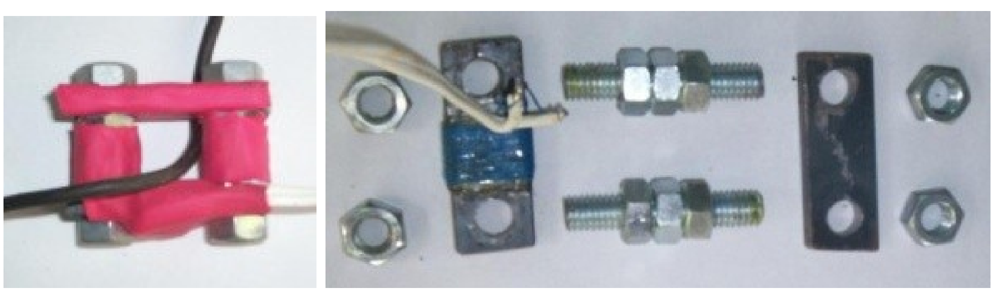

Figure 9: This inexpensive current transformer uses village-level technology, when connected to the microphone-input jack, it can measure AC amps (and by inference AC power)

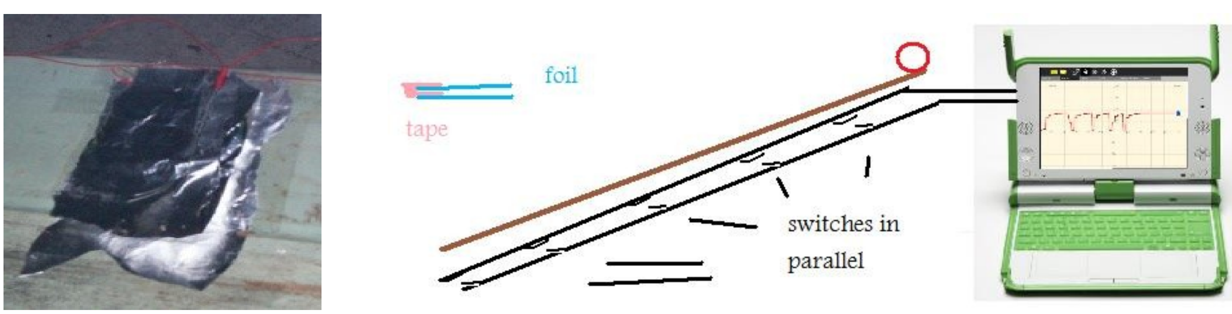

Figure 10: Simple switches (left) made out of aluminium foil used to measure acceleration on an inclined plane

5.1 Physics with XO

We work with children as young as six years of age—when their core cognitive skills and their attitudes toward learning are first developing. It is in the years of middle childhood that the brain is open for suggestions: What do I need to know? What do I want to know? For children of this age, the computer will become both a toy to play with and a tool to learn with; blurring the lines in the child's mind between play and learning and reinforcing the rewards of Papert calls "hard fun."

Students have demonstrated an emotional link to their XOs. This can be seen in the "customizations" that they have made (e.g., children add pictures, captions, stickers et al. to their XOs; some even go so far as to reprogram their desktops). We leverage this trend in our project "Physics with XO"[26]: children and adolescents, most of whom have never had access to or knowledge about computer programming, have fun creating projects that use instruments for measuring physical quantities. An underlying principle is to use minimal knowledge of programming and electronics and to design almost all of their projects directly.

An introductory project utilizes Turtle Blocks programs and the microphone to sense if sound frequency or volume exceeds a programmed threshold; in order to trigger the turtle to draw or speak. From this initial interaction, the students are motivated to build their own interactive projects, using touch sensors (simple on/off switch), LDR (light resistance transducer), or thermistor (temperature resistor transducer) attached to an audio cable.

For more complex projects, integrated sensors can be used (such as a Hall-effect sensor to measure magnetic fields) powered by the 5V DC voltage available from the XO's USB connectors. If adequate safety precautions are taken, capacitor discharge through resistors and the exponential decay of the function $V = f(t)$ can be studied.

A project that has generated great interest is programming a burglar alarm. It can be done simply by opening a contact (as described earlier) using just two wires. But we did it with a pyroelectric sensor that closes a contact between two terminals when a warm body (e.g., a person or animal) changes position. The sensor is powered by a 9-volt battery; the terminals are attached to the audio cable connected to the microphone input of the XO (See Figure 11). This minimal variation in the design sparked a great interest in all the students, they ran to begin playing with the apparatus. This reveals (again) more importantly how to design an engaging learning activity and how motivation can lead to appropriation of knowledge by students.

Figure 11: A pyroelectric sensor used for motion detection

6. Butiá project: Expanding capability with an I/O board

In September 2009 the MINA research group of the Facultad de Ingeniería, Universidad de la República, Uruguay began to work on Butiá Project[27]. The aim of the project was to create an inexpensive, open-source and versatile robotic platform to teach robotics and computer science in public schools. The objectives were to: (1) extend the XO's ability to control actuators and read sensors; and (2) reduce the asymmetries between the public and private educational systems. It builds on the Sugar software and XO hardware, in order to turn the XO into a low-cost educational mobile robot with high capability and computing power. The physical nature of the robot "allows the children to draw on their sophisticated skills and intuitions for sensing and manipulating the environments in which they live while the digital programmability allows them to turn these intuitions into formal knowledge."[28]

Behind these ideas the following concepts are present: (1) problem solving can be taken as the process by which prior knowledge is applied to new situations, far afield from that from which it originated;[29] and (2) cooperating to work together to achieve common goals. In a cooperative situation, individuals seek outcomes that are beneficial to themselves and all other group members.[30]

As seen in Figure 12 (right), the Butiá robotic kit consists of a wheeled platform upon which the XO sits and a set of sensors and actuators to which it connects. It is designed to promote experimentation, allowing the student to chose where to put sensors on the platform, using the perforated plastic pieces provided with the kit.

Figure 12: On the left, children programming the Butiá robot; on the right, a Butiá robot

Our goal is to allow students to explore and to learn the platform. In order to offer students the necessary tools for programming the robot's behavior, we needed to represent the technical details in a simple and understandable way. We chose to support the use of Sugar activities that the students in Uruguay would already be familiar with, reducing the learning gap. Consequently, Turtle Blocks was the natural choice to use for programming the robot in the younger grades (and for students who had not had previous computer science experience). An added benefit of using Turtle Blocks was its built-in support for physical interaction, allowing easy access to the XO microphone and the video camera.

6.1 Turtle Blocks Butiá Plug-in

To control the Butiá robot, a plug-in for Turtle Blocks has been developed. This software[31] allows Turtle Blocks programs to interact with the devices connected to the robot. The sensors and actuators are a set of services that have a direct mapping with Turtle Blocks blocks. These services can be available, or not, at any particular moment of time. We decided to change the color of blocks depending on the availability of the service and the physical connection of the device, avoiding the need to configure a physical port on the XO. This mechanism is user friendly and also helpful for testing: if the block is green it can be concluded that communication to the robot is working properly.

Using many sensors of the same type is a common practice in robotics. To implement this, we have the Butiá Palette automatically generate new blocks for each device connected. For example if two grayscale sensors are connected, the first one will be called "grayscale Butia" and the second "grayscale1 Butia" and so on as shown in Figure 13.

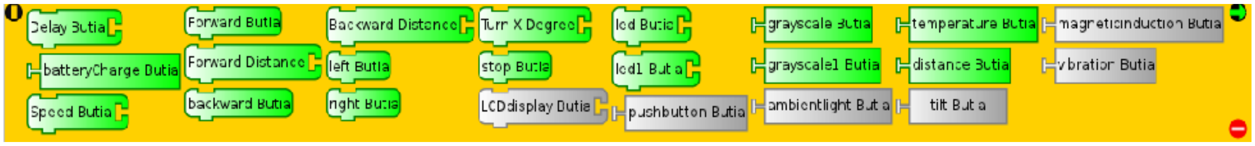

Figure 13: An instance of the Butiá Palette in Turtle Blocks

6.1.1 More Turtle Blocks plug-ins

Computer vision is a major topic of research in the field of robotics. It includes recognition of colors, shapes and object tracking. The algorithms related to these concepts require a high level of knowledge in mathematics and programming, thus they tend to be beyond the reach of beginner students of computer science. For that reason, we opted to provide support for higher-order primitives in the programming environment that would allow students to create basic robotic behaviours in a easy way. We developed a component implemented as a Turtle Blocks plug-in named FollowMe, which allows users to chose a color from an object and then obtain the coordinates of that object in the cam view. One of the classic problems in robotics is object recognition and object tracking. This plug-in enables students to work on the solution of these problems, keeping the complexity of the computer-vision algorithms encapsulated in the FollowMe plug-in (See Figure 14).

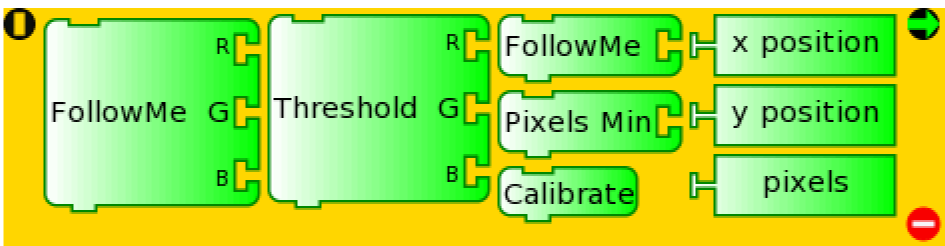

Figure 14: The FollowMe plugin

6.2 Turning XO into a mobile robot

The actuation and sensory capabilities of the XO are augmented using low-cost input/output (I/O) boards. Keeping these boards as open as possible was an important goal, allowing students to understand how they work and giving them the tools to make their own sensors, using recycled materials from old computers parts. This was done using a layered architecture to manage the complexity, combining firmware on the robot and software running on the XO computer. Students can build sensors compatible with the platform without needing a full understanding of the low-level details. Students who want to dig deeper can take advantage of the open platform. At the moment we have compatibility with two different I/O boards, Arduino that is widespread in the open source community and USB4all[32], which is based in a different microcontroller and is easy to build by hand. The architecture is designed to be easily ported to other I/O boards such as GoGo, or similar projects.

The flexible design of the Butiá project allows a wide variety of users to participate. Depending on their knowledge and interests: some students are more motivated by computer science concepts and pure robot behavior programming; some students show interest in embedded-device programming, enjoy programming of microcontrollers and access to low-level details; and some others are interested in making their own sensors and actuators.

6.3 Butiá field work

In September 2010, in a robotic workshop organized every year by the Facultad de Ingeniería, Universidad de la República, Uruguay[33], 28 Butiá robots were given to high schools in regions throughout Uruguay. Since that initial contact, we have continued to work with the high schools through a university course for students in computer science and electrical engineering. Every student in the class has to: (1) propose and develop new features for the Butiá robot; (2) give support to one of the schools that already has a Butiá robot, encouraging them to participate in new challenges; and (3) participate in robotics workshops with students of public schools or high schools in different regions of the country.

In September 2011, 50 students from high schools that have a Butiá robot have participated in robotics challenges[34]. They made use of the Butiá robot we had given them the previous year, applying what they had learned. During this two years, lectures and workshops were given for approximately 550 young students of public schools and high schools which did not receive a Butiá robot.

In November 2011 we conducted a survey of high-school teachers who had been working with the Butiá robot for at least one year. Respondents reported that work with the Butiá robot improved students abilities with XO´s and programming skills. It was also reported that students considered working with the Butiá robot to be fun and motivating; it helps students to think from different points of view and it stimulates students to do research. Teachers also reported that it is really easy to work with the robot.

The survey showed some aspects that need to be improved: teachers reported that one robot is not enough for a large group of students. They also reported a desire to build their own Butiá (we are developing a new version which can be build from scratch and may be inexpensive enough to support a one-to-one model.) They said about some problems working with teachers from basic sciences (like biology, physics) using the robot in their classes.

It was reported that programming with robots is attractive and fun for students, motivating them to learn and to find ways to apply what they had already learned in their traditional courses. Some teachers said:

> *“Motiva mucho a los alumnos el trabajar directamente sobre el hardware y ver cómo el robot actúa de acuerdo a lo programado. También pienso que desarrolla el espíritu*

de investigación del alumno, el trabajo cooperativo, el aprendizaje en base a ensayo error."
"*It is highly motivating for the students, they work directly on the hardware and see how the robot acts according to its program. It also develops the student's spirit of inquiry, cooperative work and learning based on test and error*"

"...favorecer entornos de debate, intercambio de ideas, desarrollar en los alumnos la capacidad de solucionar problemas y resolver los desafíos a los que se enfrenta"
"*... promotes environments for discussion, exchange ideas, develop in students the ability to solve problems and challenges it faces*"

6.4 Butiá conclusions and future work

Children appropriated the robot, proposing several projects on their own. Perhaps the most exciting one was a Butiá robot used to hand out end-of-year certificates[35, 36]. In the end-of-year ceremony, the director of the school put the certificate in a plastic hand adapted to the robot; the robot then followed a path along a black line on the floor to reach the place where the students were waiting.

Programming with robots allows the student to go from the abstract to the concrete (the abstract of an idea, a design, the concrete of a model built and functional). Students are the ones that choose their own projects, and each project starts from their prior knowledge, being the base for new knowledge. Each project is developed as a team, which promotes the work with others in an active and collaborative construction of knowledge.

We are working on the next version of the Butiá robot with the aim of reducing costs and simplify the electrics design. Our goal is a design in which all of the necessary electronics parts can be built by hand, enabling the experience of building for those who are interested in hardware; enforcing the idea of being full developers of the platform and not only users. We are also promoting alternative ways to build all the robot components based on the hardware re-utilization (recycling of old computer components).

7. Conclusion

Mihaly Csikszentmihályi found that people become most deeply engaged in activities that are challenging but not overwhelming[37]. Papert, in describing the concept of 'hard fun', observed that learners benefit most from activities that are 'hard' as long as they connect deeply with their interests and passions[38]. It is in their spirit of authentic engagement that we have presented our work.

David Cavallo, in his paper, "Models of Growth", argued that the "major reason for the lack of change in education is not due to lack of ideas about learning on a micro or individual level, but rather a lack of models for growth and change at a macro or systemic level.[39] Cavallo describes the pitfalls of the “replicate and scale” model, where a fully conceived change is imposed from the top down in a hierarchical fashion only to be rejected by those being force-fed, and the small pilot model, which rarely achieves any meaningful scale. We have described an approach to the use of the laptops that is not predetermined or prescribed. Rather, their use is intended to be emergent, where change is a byproduct of teachers, children, and their parents learning together, discovering new possibilities, and sharing those discoveries.

Our goal in developing the XO laptop, Sugar, and Turtle Blocks is to intervene in the approach taken to education. While we are agnostic about curricula—with the noted exception of an insistence that every child learns to program the computer—we are not agnostic about learning. We encouraged teachers and learners to nurture environments with the goal that the laptops be used for “learning to learn.”

Appendix: 30 project Ideas

In a tradition started by Papert 30-years ago, some project ideas are enumerated and described in brief:

1	Sound paint: Use the “loudness” block to change the pen size are you draw.
2	Graphing the output: Use Turtle Blocks to graph sensor output (Cartesian, polar, pie, bar, etc).
3	Measuring temperature: Use a thermoresistor to measure temperature (calibrate it to a conventional thermometer).
4	24 hours temperature: Measure temperature repeated over a 24 hour period and plot the results.
5	Measure soil moisture by using bare-wire probes.
6	Measure water salinity using copper wires.
7	Generate electricity from a changing magnetic field by wrapping wire coils around a nail and rubbing it on a magnet (measure the voltage).
8	Carbon microphone: Build a microphone from a soda bottle cap, some ground up charcoal, some foil and a rubber band.
9	Lemon battery: With a copper wire and galvanised nail, the measured voltage (try several lemons in series.)
10	Door bell/burglar alarm: When two wires touch (closing the circuit) sound a tone.
11	Measuring AC amps: Build a current transformer with copper wire.
12	Measure power

13	Measure DC amps
14	Import logged data into other activities: You can collect data to be analysed in other programs, such as a spreadsheet.
15	Logging at regular intervals: Log data based on a timer.
16	Acceleration on an inclined plane: Measure gravitational acceleration.
17	Use a light-dependent resistor to measure your pulse by placing it on your finger tip against a bright light.
18	Measure the output of a photovoltaic panel.
19	Measure the resistive-capacitive (RC) time constant of a circuit.
20	Use the XO as an audio amplifier.
21	Use the XO as an audio signal generator.
22	Build a frequency-shift (FSK) teletype for communicating at a distance.
23	Capture time-lapse photos with the camera.
24	Build a pyro-electric alarm that captures photos of intruders.
25	Build a "steady-hand" (Operation) game.
26	Explore magnetic fields with a Hall-effect sensor.
27	Build a bicycle-trip computer.
28	Build a closed-circuit TV system.
29	Build a remote doorbell.
30	Your idea goes here...

References

1. *One Laptop per Child* Home Page (http://laptop.org)
2. *Arduino* Home Page (http://www.arduino.cc/)
3. *Sugar Labs* Home Page (http://sugarlabs.org)
4. *Licenses* – GNU Project – Free Software Foundation (FSF) (http://www.gnu.org/licenses/licenses.html#GPL)
5. F. Martin, *Robotic Explorations: A Hands-On Introduction to Engineering*, Prentice Hall (2000)
6. *FIRST Robotics Competition* Home Page (http://www.usfirst.org/roboticsprograms/frc)
7. *Physics I: Classical Mechanics with an Experimental Focus*, MIT Open Courseware (http://ocw.mit.edu/courses/physics/8-01x-physics-i-classical-mechanics-with-an-experimental-focus-fall-2002/)
8. Dewey, J. *Experience and education*. (New York, NY Touchstone, 1938).
9. Papert, S., Teaching Children to be Mathematicians vs. Teaching about Mathematics," *Artificial Intelligence*, Memo 249, pp. 3–4.
10. Solomon, C., Minsky, M., Harvey, B. *LogoWorks: Challenging Programs in Logo*, Computing Mcgraw-Hill (November 1985).

11. Resnick M, Berg R and Eisenberg M: ‘Beyond black boxes: bringing transparency and aesthetics back to scientific investigation’, *Journal of the Learning Sciences* **9:**1, pp 7–30 (2000).
12. Rogoff B: ‘Observing sociocultural activity on three planes: participatory appropriation, guided participation and apprencticeship’, in Wertsch J V, Rio P D and Alvarez A (Eds): *Sociocultural Studies of the Mind*, Cambridge University Press, Cambridge, pp 139–164 (1995).
13. *Software Freedom Conservancy* Home Page (http://sfconservancy.org)
14. *XO Hardware specification* (http:wiki.laptop.org/go/Hardware)
15. *Mindstorms* LEGO NXT Home Page (http://mindstorms.lego.com)
16. *WeDo* LEGO Education Home Page (http://education.lego.com)
17. *GoGoBoard* Home Page (http://www.gogoboard.org/)
18. *Python* Home Page (http://python.org)
19. *Turtle Blocks* Home Page (http://wiki.sugarlabs.org/go/Activities/TurtleArt)
20. Abelson H and diSessa A: *Turtle Geometry,* The MIT Press (1980).
21. Harvey, B. *Computer Programming Logo-style,* The MIT Press.
22. *Turtle Art* Home Page (http://turtleart.org)
23. *GameMaker* Home Page, Yoyo Games (http://www.yoyogames.com/gamemaker)
24. *Plug-in* Wikipedia (http://wikipedia.org)
25. *Box-2D* Home Page (http://box2d.org)
26. *Física con XO* Home Page (http://sites.google.com/site/solymar1fisica/fisica-con-xo-investigacion-)
27. *Project Butiá* Home Page (http://www.fing.edu.uy/inco/proyectos/butia)
28. R W Picard, S Papert, W Bender, B Blumberg, C Breazeal, D Cavallo, T Machover, M Resnick, D Roy and C Strohecker, ‘Affective learning — a manifesto’**,** *BT Technology Journal* **22:**4 (2004).
29. Villela, J. *Ideas para enseñar a través de problemas*, Ediciones Espartaco, Uruguay, 2006.
30. Johnson, D.W., Johnson, R.T. y Holubec, E., *El aprendizaje cooperativo en el aula,* Buenos Aires, Paidós, 1999.
31. *TurtleArtButiá* Home Page (http://activities.sugarlabs.org/es-ES/sugar/addon/4434)
32. *USB4all* Home Page (http://www.fing.edu.uy/inco/proyectos/butia/mediawiki/index.php/Usb4all)
33. *Sumo.uy* Home Page (http://www.fing.edu.uy/inco/eventos/sumo.uy/)
34. ChallengesSumo2011 Home Page (http://www.fing.edu.uy/inco/proyectos/butia/mediawiki/index.php/Review)
35. Butia Student Project (http://www.youtube.com/watch?v=Qvsp4dTU244)
36. Butia Student Projects (http://www.youtube.com/watch?v=LD3C6fw-RqE)
37. Csikszentmihályi M: *Finding Flow: the psychology of engagement with everyday life*, HarperCollins (1997).
38. Papert S: *The Children’s Machine*, Basic Books (1993).
39. Cavallo, D., ‘Models of Growth’ *BT Technology Journal* **22:**4 (2004).