Erasmus+

Experimental Junior High School of the University of Macedonia (GREECE)
Zespół Szkół w Żychlinie (POLAND)
Colegiul Tehnic "Mihai Bravu" (ROMANIA)
I.E.S. Luis de Góngora (SPAIN)
Hüseyin Okan Merzeci Anadolu Lisesi (TURKEY)

Teaching Approaches to Robotics in Secondary Education

Aikaterini Chatzifoteinou

TEACHING APPROACHES to Robotics in Secondary Education

Aikaterini Chatzifoteinou

A description of the educational material and techniques implemented in “e-Robocops”, the European Erasmus+ Robotics Club

ISBN 978-960-99808-2-1

Author:

For the Experimental Junior High School of the University of Macedonia (GREECE)
Aikaterini Chatzifoteinou

Editors:

For the Experimental Junior High School of the University of Macedonia (GREECE)
Eleni Mouzoura (School Director), Maria Maheridou (Machairidou), Efthimios Mavrogeorgiadis

Creative & Digital Processing:

For the Experimental Junior High School of the University of Macedonia (GREECE)
Maria Maheridou (Machairidou), Aikaterini Chatzifoteinou

Project Manager:

For the Experimental Junior High School of the University of Macedonia (GREECE)
Maria Maheridou (Machairidou)

Cooperators:
For the *Zespół Szkół w Żychlinie (POLAND)*
Iwona Kciuk

For the *Colegiul Tehnic "Mihai Bravu" (ROMANIA)*
Gabriela Bojanopol

For the *I.E.S. Luis de Góngora (SPAIN)*
César Morales Pérez

For the *Hüseyin Okan Merzeci Anadolu Lisesi (TURKEY)*
Hatice Utaş

"Our School - My Future, ESL project"
URL: http://www.osmf.eu

Experimental Junior High School of the University of Macedonia
[Peiramatiko Gymnasio Panepistimiou Makedonias]

Strebenioti Area, Thessaloniki GR-567 01
Greece

tel.: +30 2310 587282
fax: +30 2310 587282

E-mail: mail@gym-peir-uom.thess.sch.gr
URL1: http://www.peiramak.gr
URL2: http://gympeir.uom.gr

ISBN 978-960-99808-2-1

Education is not preparation for life...

...education is life itself

Jonh Archibald Dewey
(1859-1952)

Contents

Preface

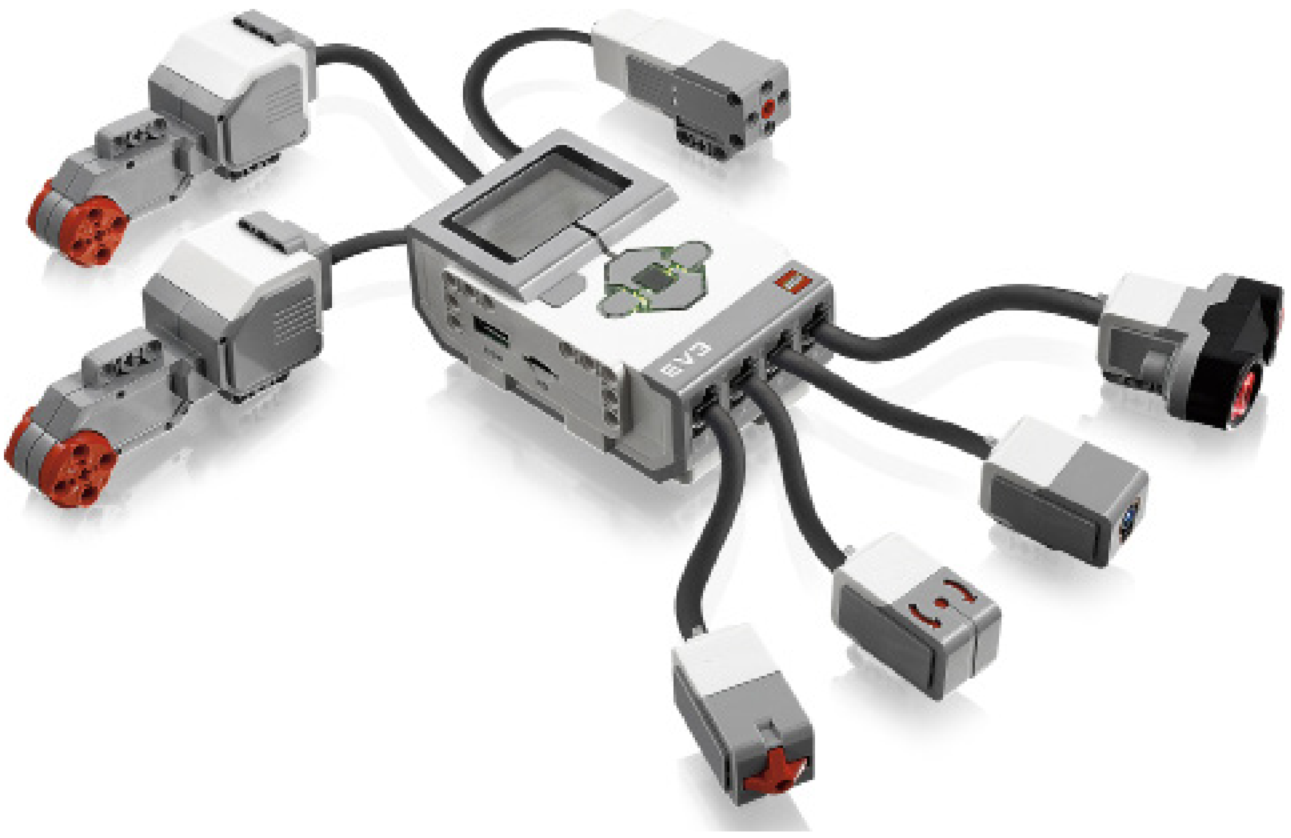
The beginnings of Educational Robotics date back to the 80's and are based on the Constructionism Theory of the MIT professor Seymour Papert (Papert, 1980). As mentioned by Watters, 2015, the first working prototype of a robotics processor for educational purposes was constructed at MIT Media Laboratory by Martin et al. (1987). Since then, a variety of educational robotics platforms has been developed and a significant amount of educational research has proved the strong pedagogical advantages of using such platforms (e.g. Papert & Harel, 1991, Erwin et al., 2000, Jonassen, 2000, Denis & Hubert, 2001, Goff & Vernon, 2001, Wang, 2001, Hacker, 2003, Dagdilelis et al., 2005, Resnick & Silverman, 2005, Ringwood et al., 2005, Staszowski & Bers, 2005, Isela & Mota, 2007, Sullivan, 2008, Nuget et al., 2009, Papanikolaou & Frangou, 2009, Datteri et al., 2012, Demetriadis et al., 2012, Mikropoulos & Bellou, 2013). The main reason why the interest of the educational community in robotics has increased so rapidly lies on the fact that, through solving robotics challenges, students are forced to use a variety of skills that they would not have the opportunity to practice in any other school subject. Most importantly, these skills are acquired in a very pleasant, creative and imagination-stimulating manner that differs widely from classical educational techniques. When dealing with robotics activities, students find it particularly attractive when they solve problems that appear in everyday life (such as developing a household coffee machine or an anti-theft alarm system). The connection of robotics to everyday life provides a very strong motivation, even to students that have little or no interest at all in the school curriculum in general, thus making educational robotics a very powerful tool for preventing early school leaving.

The work at hand is the product of many years of experience in teaching ICT, Programming and Science in Secondary Education through Educational Robotics. Multiple educational techniques for different robotics platforms, programming environments and students' level of expertise are presented. All these techniques have been applied with great success in teaching Robotics to students from 12 to 17 years old, in the Robotics Club of the Experimental Junior High School of the University of Macedonia in Thessaloniki that was founded by the author in 2011 and has been running uninterruptedly ever since.

INTRODUCING ROBOTICS

An introduction to LEGO Mindstorms EV3

Introduction: A very suitable tool for introducing robotics to students with no previous experience is the educational package LEGO Mindstorms EV3. This package is mostly suitable for 12-15 years old students, but can be also used with great success with older and younger students (aged 10 to 17). As shown in the photo below (LEGO, 2015), the package contains a brick-processor, two large motors, a medium motor, two touch sensors, an ultrasonic sensor, a color/light sensor and a gyro sensor. It is also accompanied by a wide variety of connecting and building materials. A suitable object-oriented, optical, block-like programming software is provided for free to the Mindstorm users.



Overview & Purpose: In this first contact with robotics, students will learn to recognise the various Mindstorms EV3 items and what they are used for. They will also familiarise themselves with the brick's screen and will take a first look at the Mindstorms accompanying software. We must note here a very basic concept of Educational Robotics that will follow all activities in this booklet: Team Work. All the Robotics activities that follow are designed for students working in groups of 2-3 persons. Good collaboration within a group is by itself a very important objective in every Robotics teaching and a prerequisite for the success of every robotics project. Throughout all robotics activities, students will learn to cooperate, compromise, tolerate diversity, listen and be patient. All of the above skills, which are obtained through group work that aims to complete the robotics activities, will help the students in their future personal improvement, socialization and career. For this reason, good collaboration in teams is a primary objective for all the activities in the present as well as in the following sections.

Teaching Methods: Demonstration, Brainstorming, Discussion, Hands-on activities, Investigative Learning, Trial and Error, Group work, Collaborative learning

Duration: 3 hours

Objectives/targets

At the end of the introductory lesson the students will be able to:

1) Describe each Mindstorms EV3 part and its function.
2) Connect the various elements to each other and to the Processor-Brick.
3) Navigate through the menus on the Brick's screen and read the values of the different connected sensors.
4) Make a simple vehicle with wheels motioned by the two large motors.
5) Recognise the basic features of the Mindstorms software environment.

6) Make a very simple program and download it to their robot in order to execute it.

Materials & Resources: Computers with Internet connection and the Mindstorms EV3 software installed, as well as all the equipment of the LEGO EV3 package.

Activities

Activity 1: Get to know your robot!

The various parts of the EV3 equipment and how they are connected to the EV3 Brick are first demonstrated to the students by the teacher. Before the teacher explains what each part does, students guess their function through a brainstorming activity. After everything is clarified by the teacher, a worksheet with the photo of the EV3 parts is distributed to the students and the students are asked to identify them and explain what they do. They are also asked to answer the following questions on their worksheet:

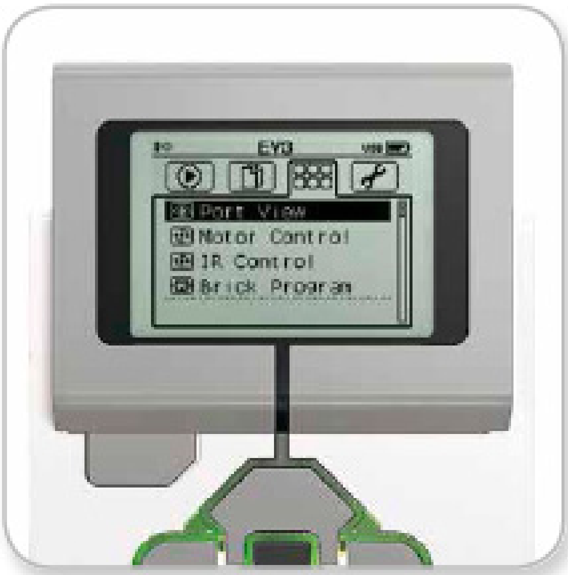
i) What do we connect to the A, B, C, D ports of the EV3 Brick?

ii) What do we connect to the 1, 2, 3, 4 ports of the Brick?

iii) Can our robot actually see?

Activity 2: Investigate your surroundings with the EV3 sensors!

The students are asked to connect the color/light sensor to Port 1 of the EV3 and use the Brick-menu shown in the image below to note the value of the sensor when we check: a) Color, b) Reflected Light, c) Ambient Light. The teacher explains the difference between the three functioning modes of the color/light sensor.



Then s/he asks the students to approach their color/light sensor within a 1cm distance from a coloured object and complete the following table on their worksheet:

Object's Color	Sensor Value: Color	Sensor Value: Reflected Light	Sensor Value: Ambient Light
White			
Yellow			
Green			
Blue			
Red			
Black			

When all groups have completed their tables, the students announce their findings in plenary session and discuss how the color/light sensor measures light intensity.

Then the teacher asks the students to replace the color/light sensor with the touch sensor and, working in a similar way as before, write down the different states that the sensor reads, depending on whether we press or release the touch sensor's button.

The same procedure is followed with the ultrasonic and the gyro sensors, with students answering worksheet questions on what exactly these sensors measure.

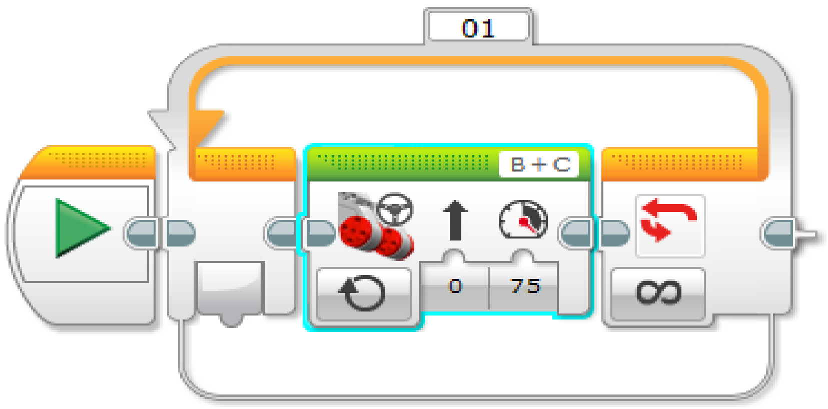
Activity 3: Build your first robot!

The students are asked to build their first robotic vehicle with wheels motioned by the two large motors, following the instructions in the URL below:

http://robotsquare.com/wp-content/uploads/2013/10/45544_educator.pdf

Activity 4: Make your robot move!

The teacher makes a short introductory tour to the Mindstorms software. After showing the basic menus and software features, s/he connects the robot to the computer through the USB cable and shows how the software recognises the various robotic parts. S/he then demonstrates to the students how they can make the following simple program, which makes their robot continuously move forward. The students can easily download this program to their robot and run it.



Assessment

Students' assessment is based on worksheet correction and direct feedback during the activities. The teacher also observes the students' group work and assesses their final robotic construction.

An introduction to programming structures

Introduction: Acquiring programming skills is one of the most demanding ICT subjects. Through the process of struggling to accomplish a robotic challenge, the students manage to regard programming as the means to achieve their goal. Thus, programming becomes a necessity and they eventually end up experientially learning basic and advanced programming skills with very little effort and without even realising the scientific importance of the structures they use. As far as programming is concerned, the teacher can watch the educational "miracle" of Game Based Learning taking place - through robotics activities - in front of his/her eyes!

Overview & Purpose: With these activities the students are introduced to the basic concepts of programming, such as the Sequence and the Loop Structures.

Teaching Methods: Guided discovery, Trial and Error, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Realise that motor rotation is proportional but not equal to the rotation of the vehicle.
2) Create a Sequence Structure in the Mindstorms EV3 programming environment.
3) Create a FOR Loop Structure in the Mindstorms EV3 programming environment.
4) Describe why and when we should use the FOR Loop Structures.

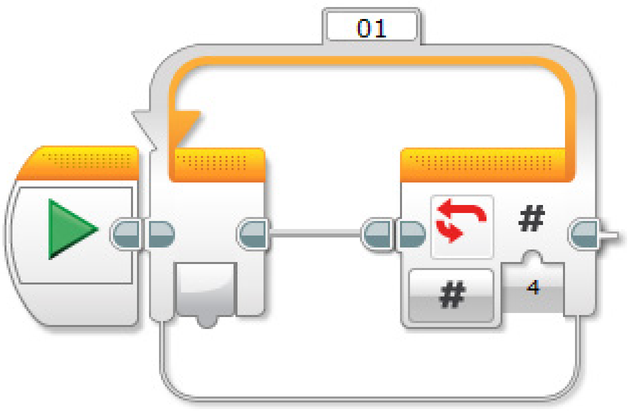
Materials & Resources: Computers with the Mindstorms EV3 software installed and the robotic vehicles created by the students in the previous lesson.

Activities

Activity 1: Moving on a square

Using a worksheet, the students are asked to use the Move Steering block from the green menu of the Mindstorms software, as many times as needed, in order to program the robotic vehicle created in the previous lesson to move on a big square. When prompted with the question: "How many blocks will you need?" the students must answer "8 blocks", since they need 4 blocks for their robot to go straight and 4 blocks to make it turn right by 90 degrees each time. The teacher explains to the students that this sequence of 8 blocks as well as any other sequence of blocks that are connected in a row (one after the other) comprises a Sequence Programming Structure. All the teams implement the square program, attempting various numbers of motor degrees in order to make their vehicle turn at right angles each time and trying to make their robot move on a proper square.

Then, students are asked to tackle the following worksheet question: "What if we wanted our robot to move on a square half the size of the previous one? How many blocks should we alter and how?". This makes the students realise that if they want to halve the size of the square, they should halve the motor degrees of all 4 blocks that move the robot forwards. As they attempt to change the motor degrees in 4 out of the 8 blocks of the sequence, the students realise that errors are much more likely to creep in, when altering a sequence programming structure.



Activity 2: Using a Loop Block

Now the worksheet guides the students to repeat the same mission using the loop block and selecting 4 repetitions at the lower right corner of the block, as shown in the image on the left.

When prompted with the worksheet question: "How many blocks will you need to place inside the loop?" the students must reply "2 blocks, one to go straight and one to turn right by 90 degrees", thus realising that using a FOR Loop structure, the length and complexity of our program decreases enormously.

At the next worksheet question: "And, if now you wanted to do the small square, in how many blocks would you have to make changes?" the students will soon realise that they must reply "in one" since they only need to make changes in one block!

At the final question: "If you want your robot to repeat a square 10 times using only the Sequence Structure, how many blocks should you use?" the students realise the absolute necessity of using a FOR Loop Structure, since the answer is 80!! While at the question "And, if you want to do the same thing with a Loop Structure, how many blocks should you use?" the answer is only 3!!

After all of the above they find no difficulty in replying to the question: "So, why is the Loop Structure useful?".

As a consolidation activity, the teacher may assign the following:

Activity 3: Move as you like!

Using the Loop structure, try making your robot move on other regular polygon shapes, such as a hexagon, or an equilateral triangle.

Assessment

Student assessment is based on worksheet correction and direct feedback during the activities. The teacher also observes the students' group work and assesses the execution of their programs on their robotic vehicle.

Sensor values and the Repeat-Until programming structure

Introduction: Performing a continuous operation until a robot sensor detects a change in the environment, is a very common event in Robotics and Automation. On the other hand, when teaching classic programming languages, the Repeat-Until Loop Structure is one of the most difficult to comprehend structures. With the Mindstorms EV3 programming software, the Repeat-Until Structure is implemented with one block only and the nature of the robotics problems involving the continuous reading of sensor values and the robot's reaction when a specific sensor value is detected, makes the use of the Repeat-Until Structure a natural consequence.

Overview & Purpose: With these activities the students are introduced to embedding sensor values into the Repeat-Until Loop Programming Structure in a very comprehensible manner.

Teaching Methods: Guided discovery, Trial and Error, Group work, Collaborative learning

Duration: 3 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Describe the function of a Repeat-Until Loop Structure and when they should use it.
2) Embed a robot sensor value into a Repeat-Until Loop Structure.
3) Use the appropriate robot sensors to locate objects.
4) Create robotic constructions to move objects.
5) Transfer motion from one rotation axis to a perpendicular one using gears.
6) Improvise with robot construction.
7) Use and program two sensors at the same time.

Materials & Resources: Computers with Internet connection and the Mindstorms EV3 software installed and the robotic vehicles created by the students in the previous lessons together with all EV3 equipment.

Activities

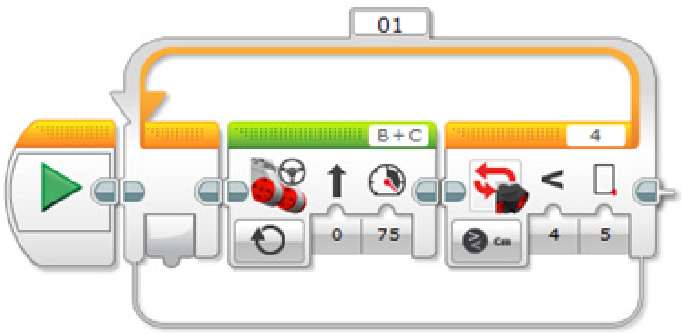
Activity 1: A robotic construction designed to move objects

Following the instructions found at http://robotsquare.com/wp-content/uploads/2013/10/45544_educator.pdf, the students are asked to construct the part of the Educator robot that can catch and carry away things. The design uses gears to transfer motion from the rotation axis of the medium motor to a perpendicular axis that lifts and lowers a bar to catch items. Thus, the students can learn how to make complex constructions with gears and transfer motion.

Activity 2: Moving pre-positioned objects

First, the students are asked to program the robot they have created to move straight until it reaches an object in a prior known position, take this object and bring it back to its base.

After all teams accomplish this task, they are asked to answer the question in the following activity.



Activity 3: Locating randomly positioned objects and moving them

Question: "What happens if we don't know the exact position of the objects we want to move in advance (as long as they are straight ahead)? How will our robot find where the objects are?". The students can use either the color sensor or the ultrasonic sensor to solve this problem, as long as they embed it into a "Repeat-Until" loop, as shown in the image on the left. In this way, they can try to program their robot to locate an object that lies straight ahead but at a random distance.

Activity 4: Programming a robot with two sensors

As a final mission, the students are asked not only to locate an object at a random distance, but also to bring it back to their base, which is marked with a black tape-line on the floor. In order to accomplish the mission, the students will need to use two sensors, an ultrasonic one facing straight ahead to locate the object and a color/light sensor facing downwards to detect the black line that marks the robot's base-area. Thus, they have to extend the construction and the program by themselves, using two sensors at the same time.

Assessment

Students' assessment is based on group work observation, the robustness of their robotic construction and the effectiveness of all three programs.

The Selection Structure

Introduction: Making the robot detect colors can provide an excellent opportunity to teach all concepts of Selection Structures in Programming: Single-alternative, Dual-alternative as well as Case selection structures. Of course, different tasks have to be designed for each purpose, as described below.

Overview & Purpose: To introduce students to all sorts of Selection Structures by using the robot's color sensor.

Teaching Methods: Demonstration, Guided discovery, Trial and Error, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

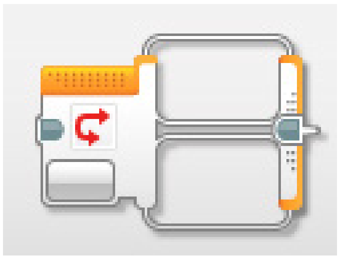
1) Apply and use a Single-alternative selection structure.
2) Apply and use a Dual-alternative selection structure.
3) Apply and use a Case selection structure.
4) Differentiate each structure and know which structure to use on each occasion.
5) Use the Selection Structure in combination with the robot's sensor values.
6) Embed one Structure into another.

Materials & Resources: Computers with the Mindstorms EV3 software installed and the robotic vehicles created by the students in the previous lessons, together with all EV3 equipment.

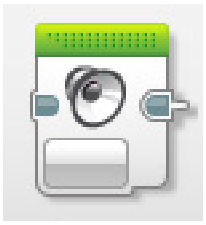
Activities

Activity 1: Single selection structure

The students are asked to use the Switch block from the orange menu of the Mindstorms EV3 programming environment (shown below) in order to make their robot say "red" if the color sensor detects a red object. The teacher demonstrates how the Switch command can get input from the various EV3 sensors and respond to specific sensor values.



He/she also guides the students to use the Sound block (shown below) and select Red from LEGO Sound Files -> Colors



When trying their program on the robot, the students will soon realise that it does not work because they haven't put their switch block into an unlimited loop, so their switch command will be executed only when, without them realising it. The teacher's help is crucial at this point, since s/he can point out to the students the moment the switch command is executed with the help of the animation incorporated by the software, which shows the currently executed block of the program in real time, provided of course, that the robot is connected to the computer via the usb cable while the program is executed. In this way, the students will understand that the machine's processor has its own "clock" and its own time at which it executes the programs, which is different from ours. This is how automation works and this is why the unlimited loop block is so important in making automatic devices.

Activity 2: Dual selection structure

The above activity is extended by asking the students to alter the program of Activity 1 in such a way that their robot says Red when the color sensor "sees" red and makes any other noise of the students' choice on any other occasion.

Again the students execute this program with or without including the switch block in an unlimited loop block, and once more they realise the usefulness of repetition and the way automations function.

Activity 3: Multiple selection structure

In this activity the students are asked to extend the switch block in their program by pressing the plus sign on the upper left of the block (appears only in the "Measure Color" option of the switch block) so that their program can detect multiple colors and the robot say the correct color each time. In this way the students will not only experientially understand the function of the Case Selection Structure, but they will also check which colors are recognisable by the EV3 color sensor (in fact they are: black, blue, green, yellow, red, white and brown, so the students can construct a Case Switch block with up to 7 selections).

Finally, when prompted with the worksheet question: "If there were no multiple selection switch block, how could our robot detect three different colors?" , the students are forced to think that they have the option to embed one switch block into another!

Assessment

Students' assessment is based on the effectiveness of the three programs.

A simple line-following algorithm

Introduction: Line following is a function that is very often needed in robotics missions, especially in the various robotics competitions. On the other hand, thinking algorithmically is a very important ICT skill and the implementation of line-following is an excellent occasion for the students to familiarise themselves with algorithmic solutions of problems.

Overview & Purpose: Algorithmic thinking and a simple line-following algorithm using one color/light sensor.

Teaching Methods: Brainstorming, Demonstration, Trial and Error, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

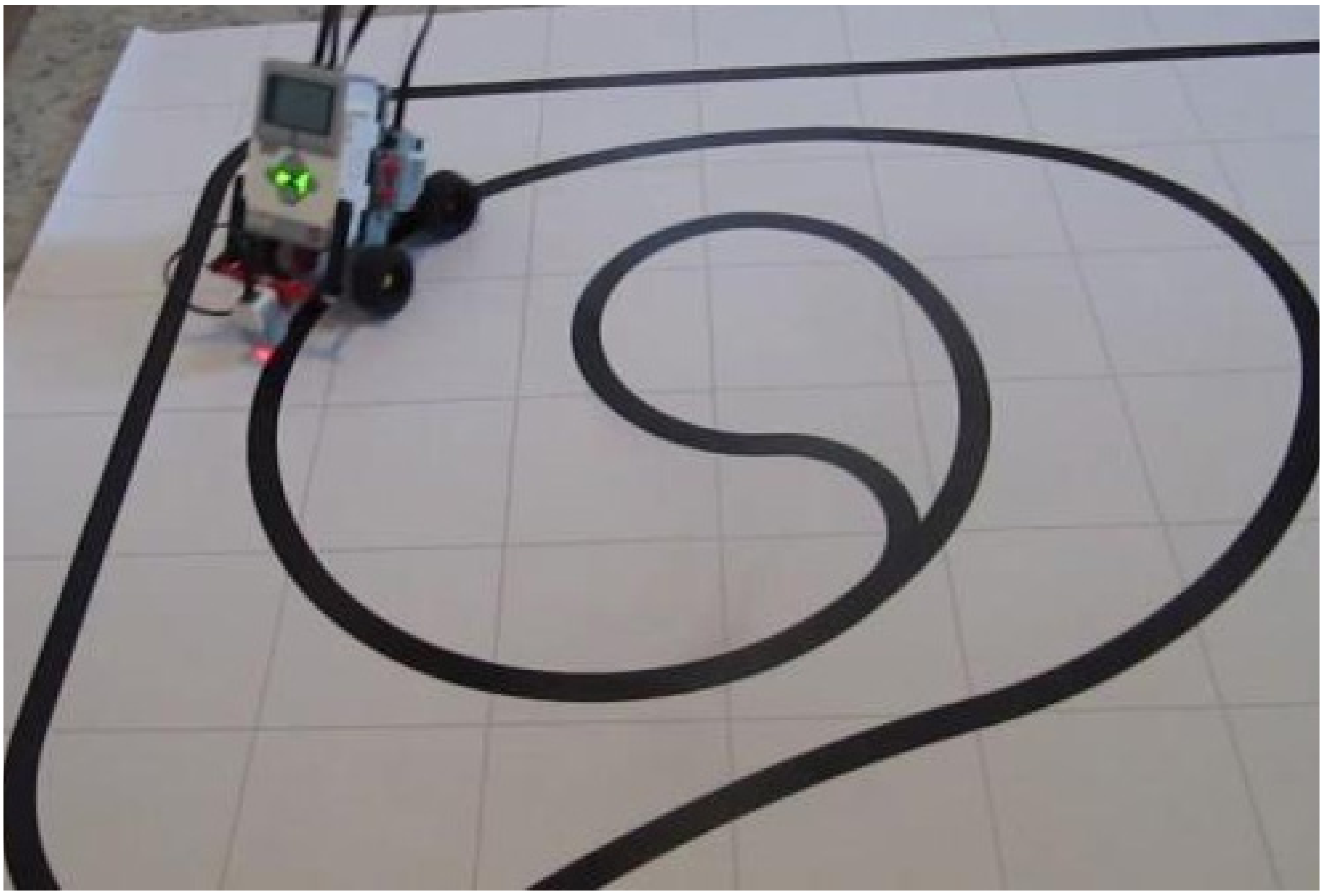
1) Make their robot follow a curved line on the floor.
2) Understand how we can use an algorithm to solve problems.
3) Consolidate how to embed programming structures.

Materials & Resources: Computers with Internet connection and the Mindstorms EV3 software installed and the robotic vehicles created by the students in the previous lessons, together with all EV3 equipment.

Activities

Activity 1: Describing the problem and finding ways to solve it

The teacher shows the following image (LEGO, 2016) and asks the students for suggestions on how we can make the robot follow the black line. After brainstorming and discussing the various ideas proposed by the students, the teacher suggests the following solution: "An easy way for our robot to follow the black line is to start with the robot's color sensor pointing, not at the line, but at the right border of the line! Then we can program the robot to continuously make small correction moves forward and left until the sensor detects black".



“When the sensor detects black, we will need to continuously make small correction movements forward and right, until the sensor detects white again, a.s.o. In this way, the robot keeps moving forward in a zigzag motion, its color sensor being   continuously kept over the black line’s right border!”

Activity 2: Making the algorithm work

The teacher asks the students to implement the algorithm described above with the aid of the Mindstorms programming software, to fit a color sensor on their robotic Educator vehicle (constructed in a previous lesson) and to make the robot follow a curved black line that the teacher has already formed on the floor with a black electrical tape.

Assessment

Students’ assessment is based on group work observation and the effectiveness of their program.

Variables: your robot's memory!

Introduction: Students often find the notion of Variables difficult to understand when taught programming the classic way. However, in the Mindstorms software, variables are visualised as suitcases, a concept that is very helpful, especially to younger ages, if they are to understand that the variable "carries" a value. Furthermore, one the lower left hand corner of the variable block, a book or a pensil appears, to distinguish reading from or writing to a variable.

Overview & Purpose: How to read from and write to a numeric variable. Using variables as the robot's memory and making the robot take decisions after measuring its environment.

Teaching Methods: Brainstorming, Discussion, Demonstration, Guided discovery, Trial and Error, Group work, Collaborative learning

Duration: 3 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Describe the purpose of using numeric variables in programming.
2) Write a sensor value on a numeric variable.
3) Read the value of a numeric variable and use it in a selection command.
4) Make the robot decide its next move depending on its environment.
5) Create complex programs with many structures embedded in one another.

Materials & Resources: Computers with the Mindstorms EV3 software installed and the robotic vehicles created by the students in the previous lessons, together with all EV3 equipment.

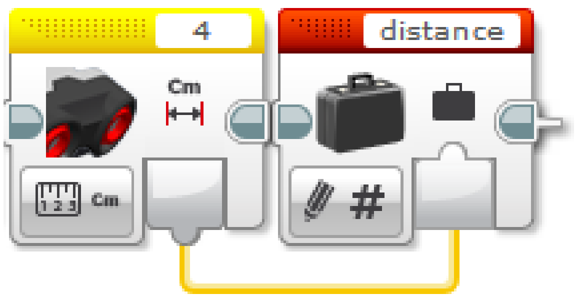
Activities

Activity 1: A robot that decides which way to go

The students are asked to make a robotic vehicle that goes forward until it reaches an obstacle. Then it stops, checks the distance of the open space on its left and on its right and decides to turn and proceed in the direction where there is more open space. In the beginning, the teacher initiates a brainstorming session so that students are forced to propose ways to solve this problem. The solution that prevails is that of fastening an ultrasonic sensor on a medium motor. Every time the robot stops moving forward, the robot should turn the sensor first left and then right in order to scan the open space available and decide on the optimal direction to proceed.

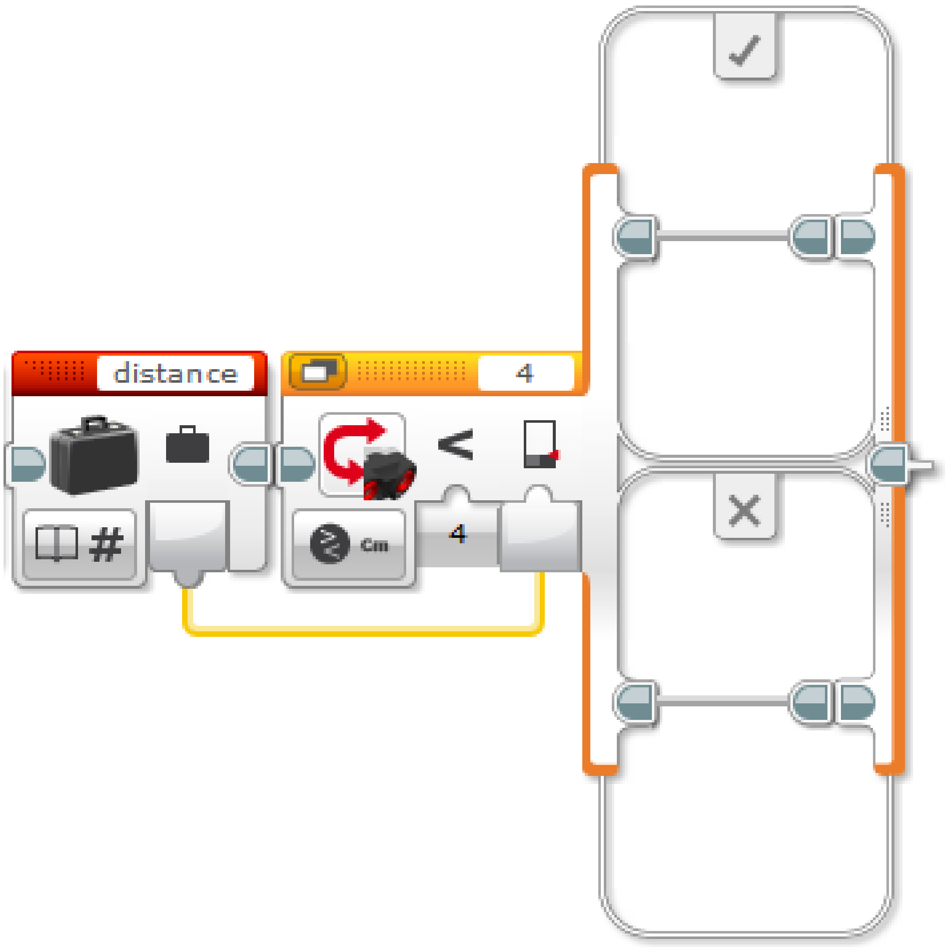
Normally, the following question arises: "How will the robot "remember" the distance measured on the left in order to compare it with that on the right, which is measured after the sensor has turned right?". The students will not have a way to reply to this question and there comes the teacher's introduction to the Variables as the robot's memory! The teacher demonstrates the variable block in the Mindstorms environment and draws parallels between a variable and a suitcase that carries things! He/she explains the different types of variables, shows how a new variable of any type can be created in Mindstorms and how it can be used in "Write" or "Read" mode. Then, s/he gives some time to the students to alter their robotic construction, adding the medium motor and the ultrasonic sensor on it, and then proceeds to the next activity.

Activity 2: Programming the "clever" robot



When programming their robot, the students have to use a structure like the one on the left in order to read the sensor value and pass it on to a numeric variable named "distance", which they must create to hold the distance of the open space on the robot's left.

Then, after the sensor turns to the right, this stored distance must be compared to the newly measured distance and the results must be used in a switch (as seen below) with which the robot "decides" to proceed to the left or to the right.



Of course, the two structures shown above have to be preceded by a Repeat-Until Loop that will move the robot forward until it reaches an obstacle, and all these blocks have to be included within an unlimited loop, so that this process is repeated. However, these two last actions must be done without any teacher hints or suggestions, since the students should be able to realise that they need these two loops, thanks to the work done during the previous lessons.

However, there is a step that the students will most possibly omit and it is better to find out "the hard way" (that is, after trial-and-error) instead of having the teacher explain it in advance: turning the ultrasonic sensor back to its original position (that is, looking straight ahead) before closing the unlimited loop that encases the whole program.

Assessment

Students' assessment is based on the robustness of their robotic construction and the effectiveness of their code.

Making a talking humanoid robot

Introduction: This is the last one of the series of introductory robotics activities for children and young adolescents with the use of LEGO Mindstorms EV3. A nice idea would be to close this series of lessons with a more impressive construction that has some anthropomorphic characteristics, mostly for motivating the students to devote further time to robotics. Constructing a small humanoid that can walk around, move its hand and talk is such an example.

Overview & Purpose: Constructing a small humanoid robot that can move and output recorded speech.

Teaching Methods: Demonstration, Guided discovery, Trial and Error, Group work, Collaborative learning

Duration: 4 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Transfer motion from one rotation axis to another using gears.
2) Use tracks instead of wheels to move the robot.
3) Upload recorded sounds to the robot and reproduce them as output.
4) Create complex robotic constructions.
5) Simulate a robotic humanoid with moving legs and hands.

Materials & Resources: Computers with microphones and speakers and with the Mindstorms EV3 and the Audacity software installed. All the equipment of the LEGO EV3 package.

Activities

Activity 1: Constructing the humanoid

Students are asked to follow the instructions for EV3RSTORM on http://www.us.lego.com/en-us/mindstorms/community/bi/ and construct the humanoid.

Activity 2: Making the humanoid walk

Students are asked to use the Move steering block appropriately, exploit their previous experience with programming motor motion, and make their robot walk.

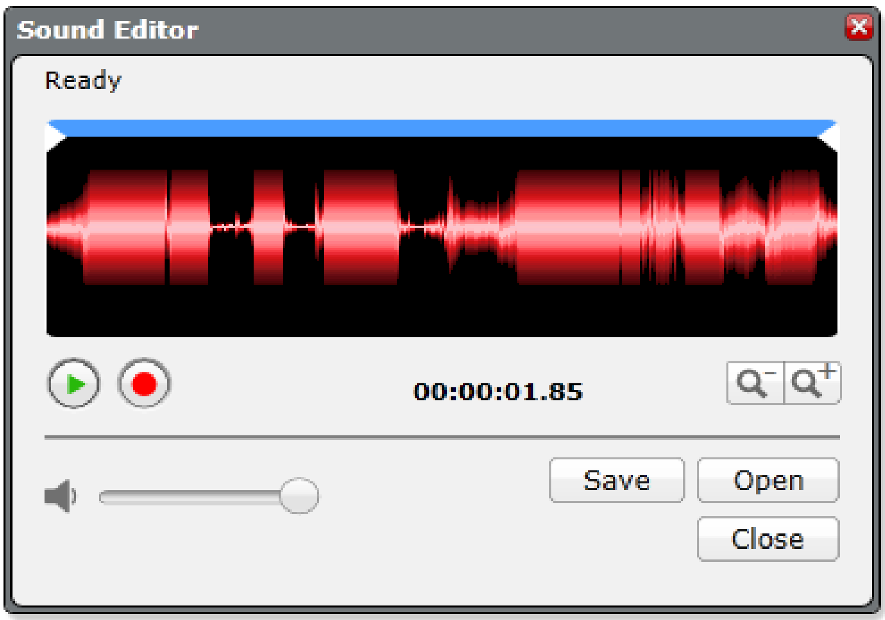
Activity 3: Moving the humanoid's hand

Students are asked to move the robot's hand by programming the motion of the medium motor.

Activity 4: Making the humanoid talk

As a final mission, the students are asked to make their robot talk. In order to accomplish this, they first have to record what the robot will say. They can do this with a separate sound editing software such as Audacity and save the sound files in wav format, or they can choose to record the robot speech through the EV3 software. In both cases, the duration of each recording must not exceed 5 seconds, the EV3 audio reproduction time. In order to upload the sounds to the EV3, guidance from the teacher is necessary. From the EV3 software menu, they must select Tools->Sound Editor (see the image on the next page). They can either do the recording and then save the sound file giving a name to it or open the wav sound file created earlier with Audacity and save it again with the EV3 software. Now, when inserting a Sound block into their program, they can select for this block to reproduce their sound, which is now contained within the "Project Sounds" list. In this way the students can make their humanoid say whatever they like! However, when recording, the volume of the sound must be as high as possible, since the EV3 playback volume label is very low.

As an extension to the above activity, the students may want to try and make their robot interact with the humans around it. For example, it could be programmed to ask a question and prompt the human user to press its touch sensor on its right shoulder if s/he wants the robot to move its hand or put a finger on the color sensor on its left shoulder, if they want the robot to dance, a.s.o. If programmed in a "clever" way, the robot may even be able to make virtual dialogues with humans. It's all up to where the students' imagination can go!



Assessment

Students' assessment is based on group work observation and the final "behavior" of their robotic humanoid!

Going further with LEGO Mindstorms EV3

A lot of work has recently been done by various authors on educational activities with LEGO Mindstorms EV3. Searching the web, teachers and students or students can find a variety of educational senarios and instructions for impressive constructions that can motivate even the most "difficult" child or adolescent! Furthermore, the EV3 processor can be programmed with more advanced languages such as ROBOTC (Robomatter, 2014). As a result, there is still so much to do in robotics with Mindstorms. Indicatively, some of the most important publications / websites where the reader can find a great amount of material and ideas to work on are worth mentioning (in alphabetical order): Bagnall (2014), Benedettelli (2013), Bratzel (2014), Francis et al. (2014), Garber (2013), Griffin (2014), Higashi & Shoop (2014), Isogawa (2014), Karch (2014), Kee (2015), LEGO (2016b), Rollins (2014), Valk (2013a, 2013b, 2014, 2015). A very interesting database of LEGO Mindstorms NXT projects that can easily be adapted to EV3 equipment is also the work of Parker (2014).

INTRODUCING AUTOMATION

Introducing Automation with Arduino and ArduBlock

Introduction: Automation is everywhere in our lives. Helping students realise how systems like our car alarm are made and giving them the opportunity to construct such systems themselves, can be the greatest motivation their school provides! The Arduino open-source electronics platform (Arduino, 2016) developed in Italy in 2005 by Barragan, Banzi, Cuartielles, Mellis, Marino, & Zambetti (see e.g. Gibb (2010), Kushner (2011) for reviews) is very suitable for an introduction to automations as well as for building low-cost robotic systems of any kind. Much research has recently been done on the multiple educational uses of Arduino. Indicatively, we should mention Igoe (2009) and Huang (2015). The educational innovation of Arduino lies mostly on the fact that it stimulates the student's imagination by enabling the incorporation of old useless electronics material (e.g. from printers, cd-rom drives, old electronic toys etc.) into new automatic robotic devices. This fact together with the enormous amount of Arduino-compatible sensors of any kind the user can find in the global market and the huge support the internet community of users and developers around the world which can provide guidance and help for almost anything one can imagine make the use of the Arduino platform a very stimulating educational choice.

Most Arduino boards incorporate Atmel's ATmega328 microcontroller, which can be programmed via USB-connection to a computer through the Arduino IDE (Arduino, 2016). The latter enables programming with a variant of the C language and includes many ready and appropriately categorized examples with various sensors and motors. Developers around the world have created ready-to-use C-libraries for all sorts of electronics equipment that can be connected to the Arduino boards. However, the C language cannot be considered the most suitable educational tool for novice programmers and younger students.

For the purposes of making the Arduino platform approachable to younger students (e.g Junior High School or even older Primary School classes), many visual programming environments have been developpped over the past few years which can be successfully used to program the Arduino microcontrollers. Such environments are Scratch for Arduino (S4A, Citilab 2015), Minibloq (Gillig, 2014), and the web-based editor Blocklyduino (http://www.gasolin.idv.tw/public/blockly/apps/blocklyduino/index.html, Tewari, 2014), Modkit Micro (Modkit, 2016) and Ardublock (Taweili & Qichen, 2011) (see also SparkFun (w.d.a) for a review).

Some of these environments are self-sustained while others are just extensions built on top of the Arduino IDE that provide a user-friendlier block-based interface. S4A is a widely used environment but has the disadvantage that it only runs on the USB-connected computer and the code cannot be uploaded onto the ATmega328 controller. As a result, all S4A programs must run with the board connected to the PC. In this section we will present educational scenarios using Ardublock, since Ardublock incorporates a wide variety of sensors and other electronics and has the extra advantages of: a) running within the Arduino IDE, so that the novice programmer can see how his/her block's code is directly translated to code in C language, thus getting emotionally and cognitively prepared for professional programming environments b) being uploadable to the Arduino board, since it's actually C-code and can run independently.

Overview & Purpose: In this first approach to the Arduino board and automations, the students will learn to recognise the different kinds of Arduino input and output, digital and analog pins as well as a variety of Arduino-compatible sensors, buttons, buzzers, motors, LEDs and resistors that will be demonstrated to them. They will also familiarize themselves with the ArduBlock interface and the way the Arduino boards communicate with the PC and user ArduBlock commands can be uploaded to the microcontroller through the Arduino IDE. It is essential that the students work in groups of 2-3 persons in all the educational scenarios that follow.

Teaching Methods: Demonstration, Trial and Error, Hands-On activities, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Describe the Arduino header pins and various electronic material of the Arduino kit.
2) Navigate through the Ardublock environment.
3) Connect the Arduino board to the computer and upload a simple ArduBlock program.

Materials & Resources: Computers with Arduino IDE and ArduBlock installed. All the Arduino equipment that has been chosen by the teacher.

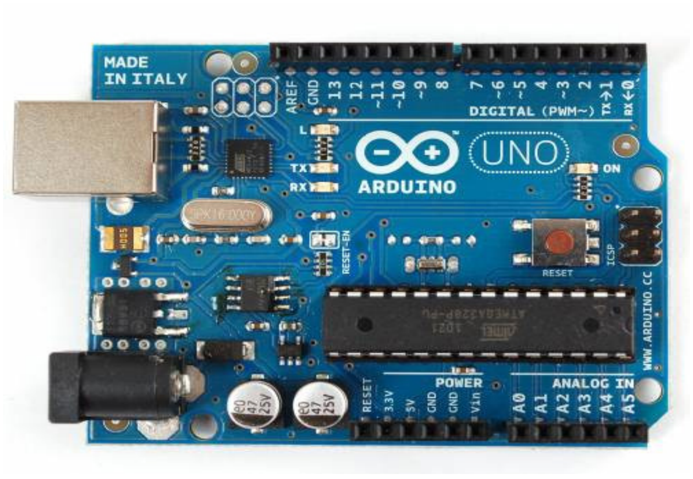
Activities

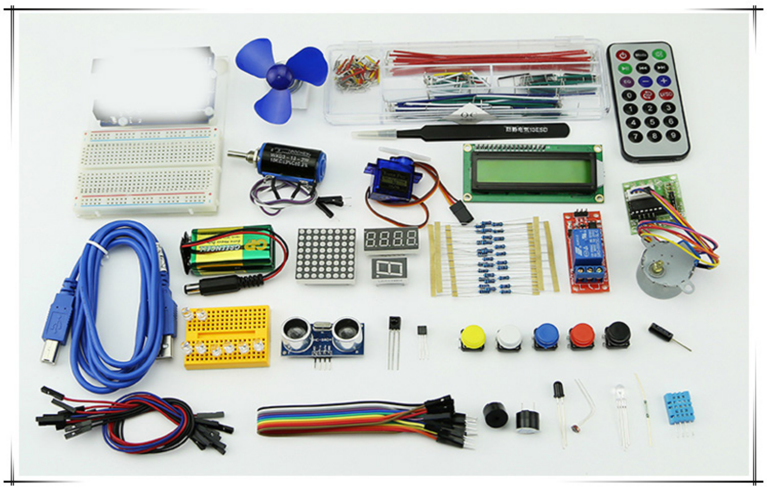
Activity 1: Getting to know the Arduino equipment

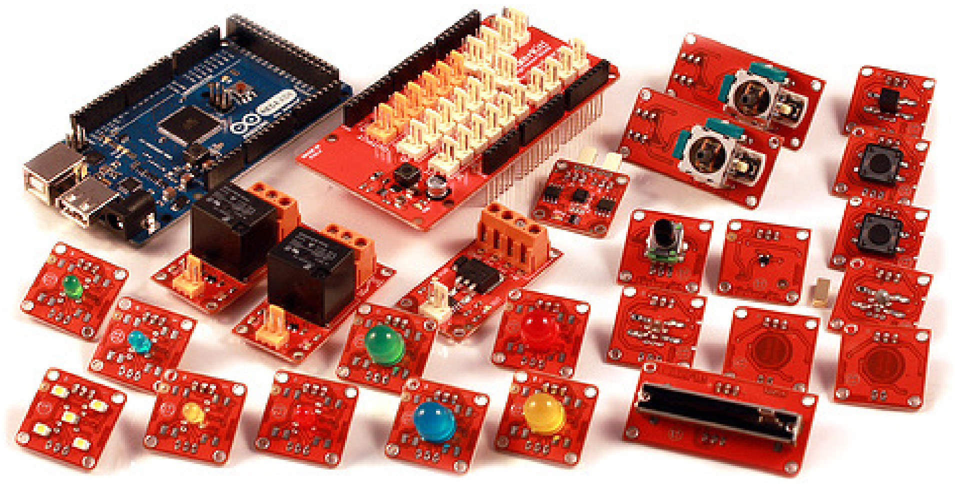
How this activity is conducted depends mainly on the kind of Arduino equipment chosen by the teacher. There is a variety of Arduino kits of different price ranges in the global market and the exact choice of Arduino board, sensors, motors etc. depends on the budget and the kind of educational projects that the teacher intends to develop with his/her groups.

For example, if one mostly focuses on automations, one should mostly supply one's groups with electronics equipment such as LEDs and resistors. On the other hand, if one focuses on low-cost robotic constructions, one should buy a set with wheels and a variety of motors in order to construct robotic vehicles similar to those constructed with a LEGO Mindstorms kit.

Sample photos of an Arduino Uno board and various Arduino kits and equipment are shown below.







No matter what type of equipment the teacher chooses to use, the most suitable educational approach is to let the students lay hands on the equipment and at the same time demonstrate the use of each material by showing relevant constructions either by the teacher or found on YouTube. Demonstrating what can be constructed with the material the students have in hand is the best motivation for them to get started!

Activity 2: Connecting the Arduino board

Before showing the students how the Arduino board can be connected to the computer, some steps have to be taken by the teacher him/herself:

a) download an install the Arduino IDE from the official arduino site (Arduino, 2016).

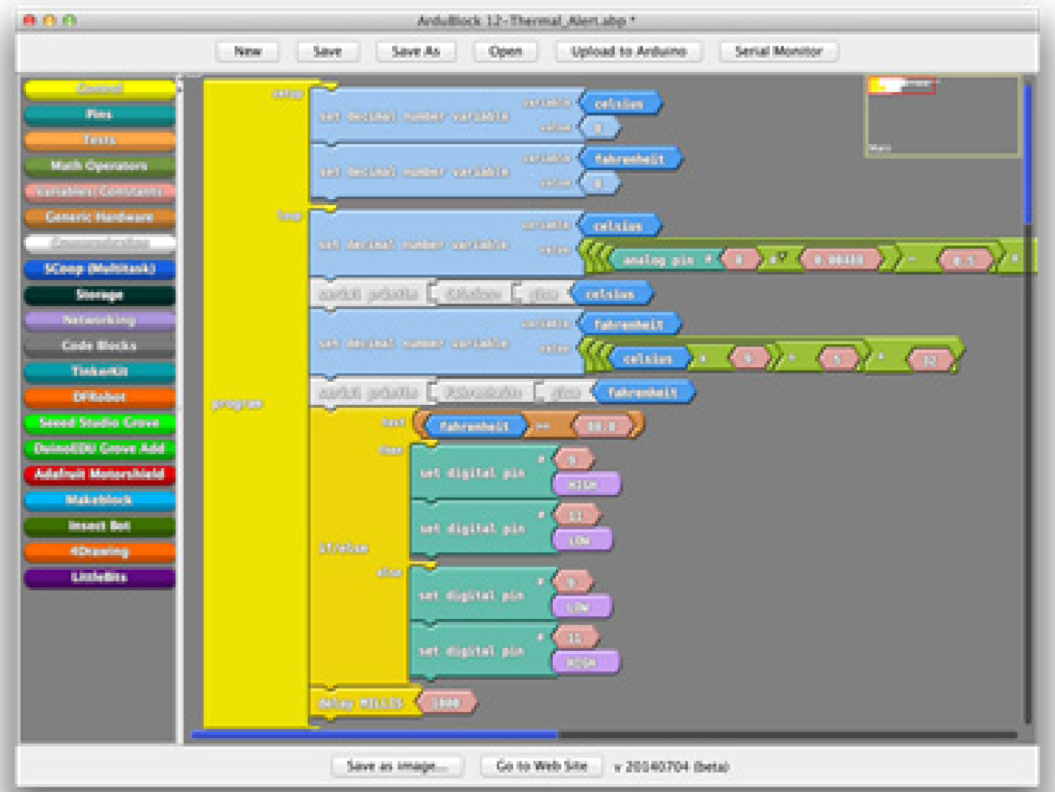
b) connect the board via USB to the computer, find the connected board and perform a driver update (if in a Windows environment) through the Device Manager of the Control Panel, in order for the operating system to recognise that the board's driver is where the Arduino software has been installed.

After these steps have been taken by the teacher, the students can be instructed to select the connected Arduino board from the Tools of the Arduino software and also choose the appropriate serial port through the same menu.

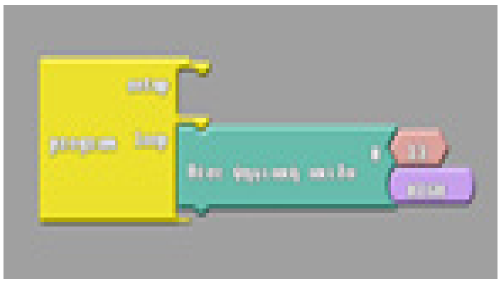
Activity 3: Opening ArduBlock and making a program

Before showing the students the ArduBlock environment, the teacher has to copy the downloaded ArduBlock Java file into the Tools folder of the Aduino subfolder within the (Windows) Documents folder. When the Arduino IDE software is executed, the students will see that a new "ArduBlock" option has appeared in the Tools menu!

The ArduBlock visual programming environment and a sample ArduBlock code are shown in the image below.



The teacher must offer a quick tour around the ArduBlock commands, focusing on the yellow and green **Control** and **Pins** menus and then ask the students to make a very simple program to turn on the board's built-in LED on pin 13 and upload it to their board through the command **Upload to Arduino**. The code is shown below in the Greek ArduBlock environment



The plain code is as follows:

loop [set digital pin 13 HIGH]

where HIGH means ON, that is, current goes to digital pin 13 and so, the LED that is connected there turns on.

The students will soon see the board's LED emit light and realise how easy it is to control their Arduino!

Furthermore, each time an Ardublock code is uploaded to Arduino, the corresponding C code automatically appears as a Sketch on the Arduino IDE screen. In this way the students can see what the "real" Arduino code looks like.

Assessment

The students' assessment is based on group work observation and how effectively they have completed Activity 3.

Creating the first circuit

Introduction: In order to be able to construct simple circuits, the teacher must offer a basic introduction to electronics. The GND and the 5V header pins of the Arduino board may be used to explain what it means to "close the circuit". The way a breadboard is constructed and how it is used to make easy connections, the reason resistors exist and the resistor color code are things that must first be explained.

Overview & Purpose: Getting acquainted with the Arduino board, LEDs, resistors, the use of breadboard and how to make a simple circuit as well as the forever loop structures in automations and the importance of the delay command in programming electronic devices.

Teaching Methods: Demonstration, Trial and Error, Hands-On activity, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

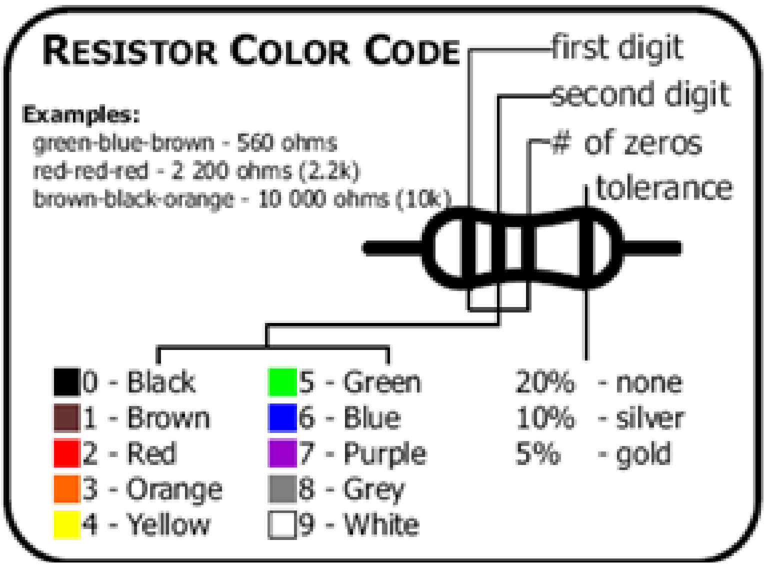
1) Distinguish the various kinds of resistors.
2) Connect a LED to the Arduino.
3) Describe and be able to use the loop and delay commands
4) Send output to a digital Arduino pin.
5) Insert comments into the code.

Materials & Resources: Computers with Arduino IDE and ArduBlock installed. The Arduino boards, breadboards, LEDs, resistors and jumper wires.

Activities

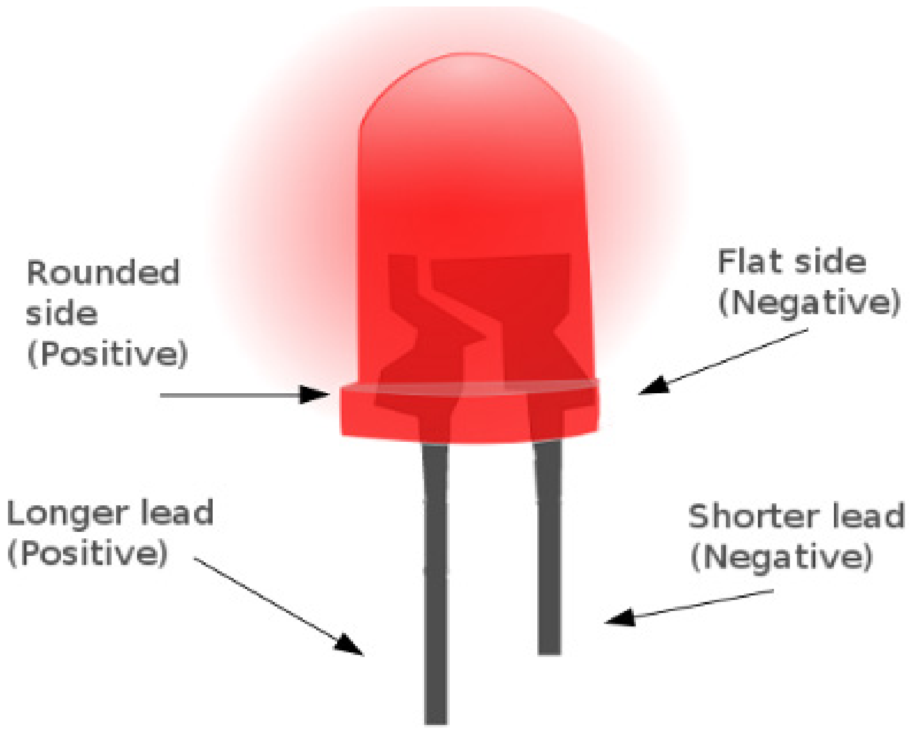
Activity 1: Getting to know LEDs and resistors

The teacher must make the students realise that a LED needs 2V to light up while the board's output is 5V. That is why a 220 Ohm resistor is necessery, in order to lower the voltage and not damage the LED. In order to find which resistor is 220 Ohm, the students must familiarize themselves with the resistor color code, shown below.



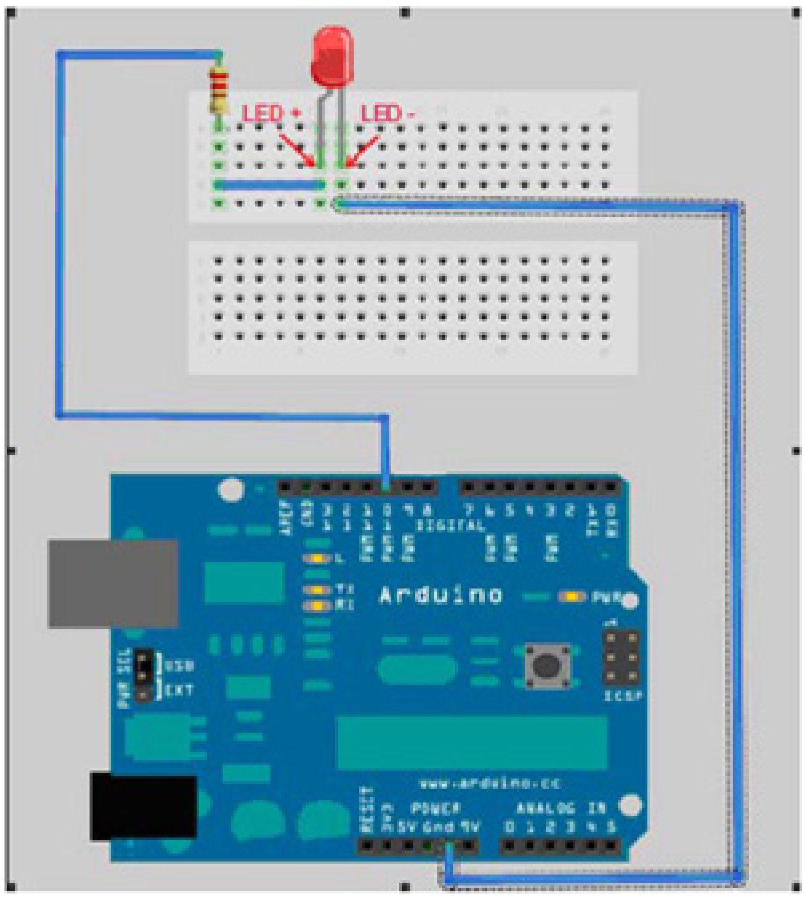
In the color code picture above we can see that a 220 Ohm resistor has red, red and brown stripes. Furthermore, the students must notice that the LED 's leads are different and note the polarity, as shown in the image below.

As a result, the longer lead must be connected to the board's 5V pin (of course, by interposing the resistor to lower the voltage to 2V) and the shorter lead must be connected to one of the GND (Ground) Arduino header pins.



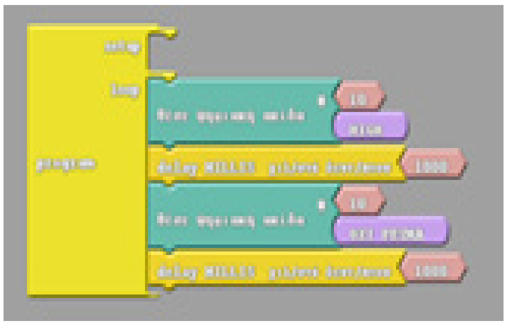
Activity 2: Constructing the circuit

Each team must be given the appropriate equipment and must be asked to construct the following circuit, with the LED's longer lead connected (via the 220 Ohm resistor) to the board's digital pin 10.



Activity 3: Flashing the LED

Now, the students are asked to write the appropriate ArduBlock code, so that the LED on pin 10 flashes every other second. This means that, digital pin 10 must receive current for 1 sec and then keep the circuit open for another sec, which can be translated into the following ArduBlock code:



The plain code is as follows:

```
loop   [ set digital pin 10 HIGH
           delay 1000 milliseconds
           set digital pin 10 LOW
           delay 1000 milliseconds  ]
```

The students must be encouraged to write comments on their code, so that later on they can remember what each block of their code does. Commenting is very easy with Ardublock, since it can be enabled for each block that is included in the program by right clicking on it!

Assessment

Students' assessment is based on group work observation and how effectively they have completed Activities 2 and 3.

Playing notes with a buzzer

Introduction: Even simple electronic equipment such as a small buzzer can instigate the student's imagination and creativity, guiven the apropriate stimulus by the teacher!

Overview & Purpose: Getting to know buzzers, note frequencies and sending analog output to Arduino header pins.

Teaching Methods: Demonstration, Guided discovery, Trial and Error, Hands-On activities, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

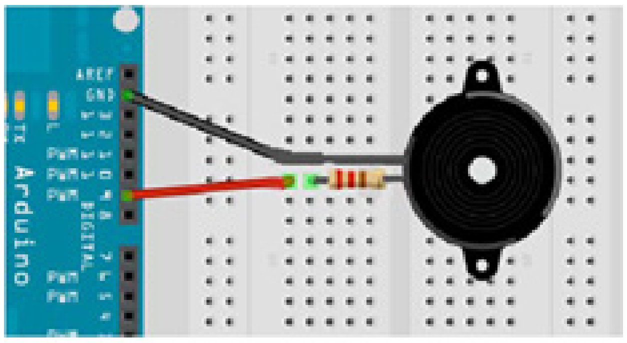
1) Make a simple circuit with a buzzer.
2) Send analog output to Arduino header pins..
3) Make a big program in Ardublock: the buzzer music application.

Materials & Resources: Computers with Arduino IDE and ArduBlock installed. The Arduino boards, breadboards, buzzers, resistors and jumper wires.

Activities

Activity 1: Connecting the buzzer

A buzzer has the same polarity as a LED and may be connected in the same way (that is, the shorter lead connected to the board's GND and the longer lead to a PWM Arduino pin, e.g. pin 3). As in the case of the LED, a 220 Ohm resistance may be interposed in the circuit, but it is not necessary). The final circuit is shown below.



Activity 2: Programming the buzzer

Now the students have the knowledge to program the buzzer to make a short sound every other second. They can make a program similar to the one they prepared to program the flashing LED, the only difference being that they must use the block from the green **Pins** menu:

set analog pin

or even better, use the block **note** instead. This block has been created specifically to play notes, since, apart from the analog pin's number and the note frequency, it also contains the duration for each note.

Activity 3: Making music

Now the teacher can ask the students to compose a small well-known piece of music (such as "Twinkle twinkle little star"!) by successively selecting **delay** blocks and **note** blocks. The only help the students need, is for the teacher to inform them about the frequency of each note of the pentagram. Such frequencies (in Hz) are given below.

c: 261	**d: 294**	**e: 329**	**f: 349**	**g: 392**	**a: 440**	**b: 493**	**c: 523**

The rest is only a matter of creativity for the students!

Assessment

Student assessment is based on group work observation and how effectively they have completed all three activities.

Pressing the button...

Introduction: Buttons are used in every automation. It will be very useful for the students to know how a button works and how it can be programmed. After this lesson, students will look at the electronic buttons that are around them from a new perspective!

Overview & Purpose: The students will learn how to make an electronic device with a button and a LED. The LED will be on for as long as the button is pressed. In this way the students will also have the opportunity to program their device using a dual-alternative selection structure.

Teaching Methods: Demonstration, Guided discovery, Trial and Error, Hands-On activities, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

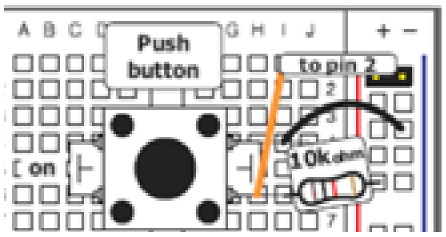
1) Make a circuit with a button and a LED.
2) Use a dual-alternative selection structure with input from the button and output to digital pins.

Materials & Resources: Computers with Arduino IDE and ArduBlock installed. The Arduino boards, breadboards, buttons, resistors, LEDs and jumper wires.

Activities

Activity 1: Making the circuit

The LED must be connected to the board's digital pin 10 together with a 220 Ohm resistor, as shown in a previous lesson, while the button will be connected to a 10 kOhm resistor, as shown in the image below.

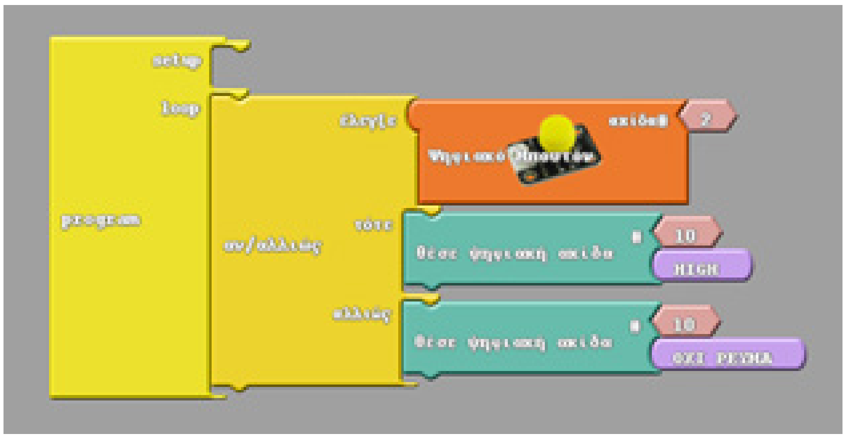


Therefore, only the two leads that are on the same side of the button (see photo below) should be connected.

The first lead should be connected to Arduino's PWM 2. The students must also connect the board's 5V output via a 10 kOhm resistor on the same breadboard line where this lead is connected. The second lead of the button should be connected to the board's GND.

Activity 2: Programming the device

Now the students are asked to program the device so that the LED is ON for as long as the button is pressed, and OFF when the button is not pressed. They must use the **if/else** block and any available button from the menus (e.g. that of the orange DFRobot menu). Their final program should look like the one below,



The plain code is as follows:

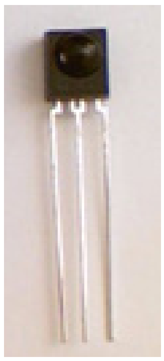
```
loop   [ if Digital Push Button 2
         then set digital pin 10 HIGH
         else  set digital pin 10 LOW  ]
```

Assessment

Student assessment is based on group work observation and how effectively they have managed to make their device work.

Remote Control Automation

Introduction: An interesting variation of the previous application that will really excite the students, is to make a remote switch using an Arduino IR (InfraRed) remote control (shown below). "Imagine we could turn on our house lights with a remote control!" - that's a suitable motivating phrase the teacher could use to get things going. This project can be easily implemented by connecting the infrared sensor that communicates with the remote control to the Arduino. Then, using appropriate programming structures, the students should be able to turn the LED ON and OFF by pressing any button of the remote control. However, the main difference from the previous project lies in the fact that, in this case, the remote control button will be pressed once and the LED will turn on and remain ON until any remote control button is pressed again.



Overview & Purpose: The students will learn to connect and use an IR remote control. Since a push of the remote control's button may lead to two different actions (turn the LED ON or OFF) depending on the LED's previous condition, two selection structures must be used, one embedded into the other. This example is a good opportunity for students to practice in constructing more complex programming structures.

Teaching Methods: Demonstration, Guided discovery, Trial and Error, Hands-On activities, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Make a circuit with an IR remote control and a LED.
2) Use a digital pin's value (HIGH/LOW) as the control condition of a selection structure.
3) Embed two selection structures into one another.

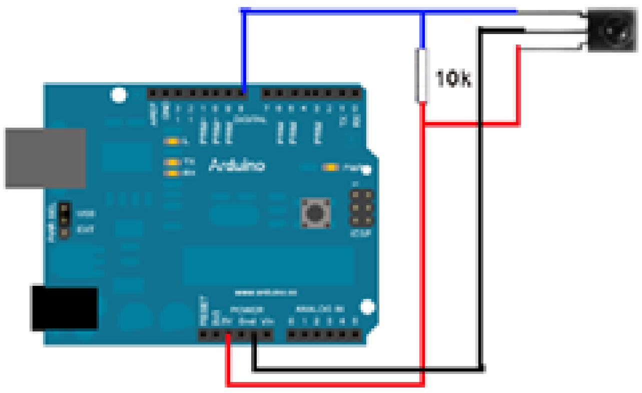
Materials & Resources: Computers with Arduino IDE and ArduBlock installed. The Arduino boards, breadboards, IR remote controls, resistors, LEDs and jumper wires.

Activities

Activity 1: Making the circuit

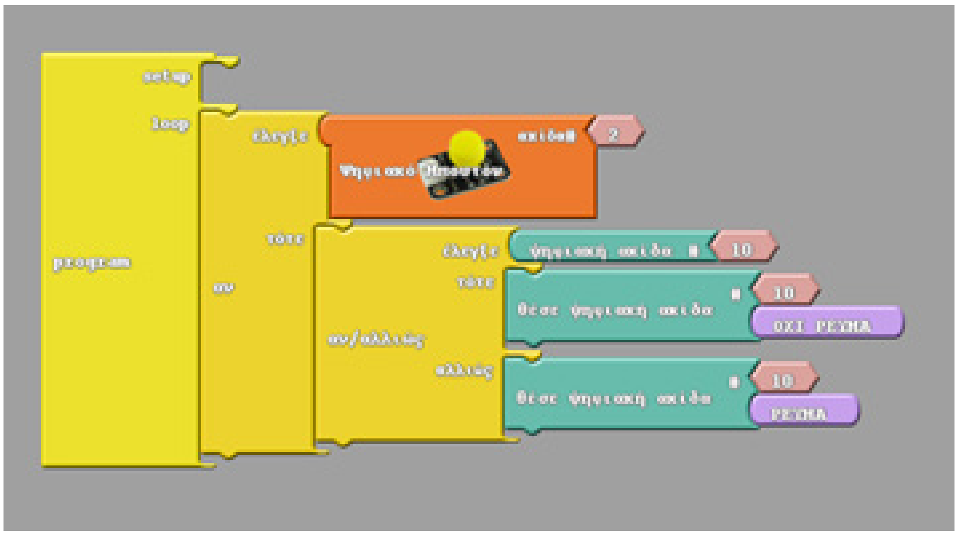
The Infrared Receiver IR-3638 (shown in the photo above) has three leads. The left lead can be connected to Arduino's PWM 2 while the right lead must be connected to the 5V ouput. However, these two leads must also be connected via a 10 kOhm resistor, similar to the one used with the button in the previous lesson. Finally, the middle lead must be con-nected to GND (see the image below).

The LED must be connected to the board's digital pin 10 together with a 220 Ohm resistor, as shown in the previous lessons.



Activity 2: Programming the device

Now the students are asked to program the device so that every time the IR Receiver detects that a remote control button has been pressed (that is, there is current on Arduino's PWM pin 2) the LED condition changes (that is, if it's ON it should be turned OFF, and if it's OFF it should be turned ON). In order to accomplish this, they must embed two **if** blocks into one another, as shown in the image below. The second if/else block must have the current value of digital pin 10 (HIGH or LOW) as a control condition.



The plain code is as follows:

loop   [if Digital Push Button 2

then if digital pin 10

then set digital pin 10 LOW

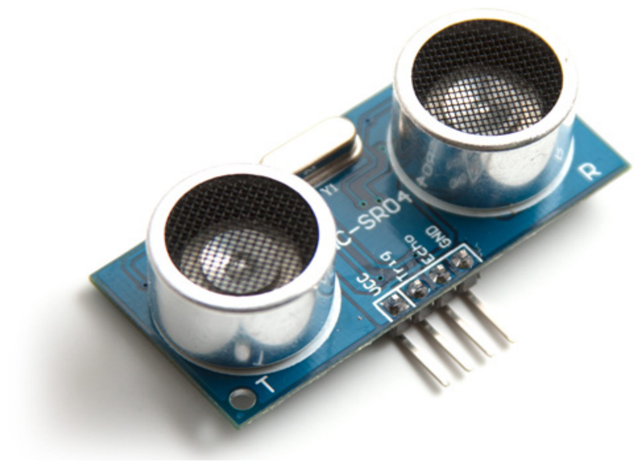
else set digital pin 10 HIGH]

Assessment

Student assessment is based on group work observation and how effectively they have managed to make the infrared device work.

Making an anti-theft alarm system

Introduction: Another interesting project that is very easy to implement with ArduBlock involves an ultrasonic sensor to detect when someone - or something - is approaching a valuable object within a user specified distance and then sound the alarm with the help of a buzzer. Most Arduino ultrasonic sensors have four leads (shown below). The Trig pin (second from the left in the picture) is used to send a sound signal. As soon as the signal is sent, a built-in procedure starts counting the time via Arduino's internal clock. Time-counting freezes when the Echo pin (third from the left) receives the sound signal back. The time measurement is devided by 2 (since the sound signal was reflected on the obstacle and then returned, so it covered double the distance) and then multiplied by the speed of sound. In this way, the ArduBlock's incorporated ultrasonic function can calculate the distance measured by the ultrasonic sensor in cm.



Overview & Purpose: The students will figure out how distance can be measured through sending sound signals and how sonar devices and radars work! They will also learn to connect an ultrasonic sensor and program it through ArduBlock. For the purposes of the project, logical testing conditions must be used. Finally, the students will familiarize themselves with the basic principles for building any sort of alarm system.

Teaching Methods: Demonstration, Guided discovery, Trial and Error, Hands-On activities, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Describe how distance can be measured by sending a sound signal.
2) Make a circuit with an ultrasonic sensor.
3) Use comparison operators in programming.
4) Describe how an alarm system works.

Materials & Resources: Computers with Arduino IDE and ArduBlock installed. The Arduino boards, breadboards, ultrasonic sensors, buzzers, resistors (optional) and jumper wires.

Activities

Activity 1: Making the alarm circuit

The ultrasonic sensor (shown above) must be connected in the following way:

VCC goes to Arduino's 5V pin

Trig goes to Arduino's PWM 2

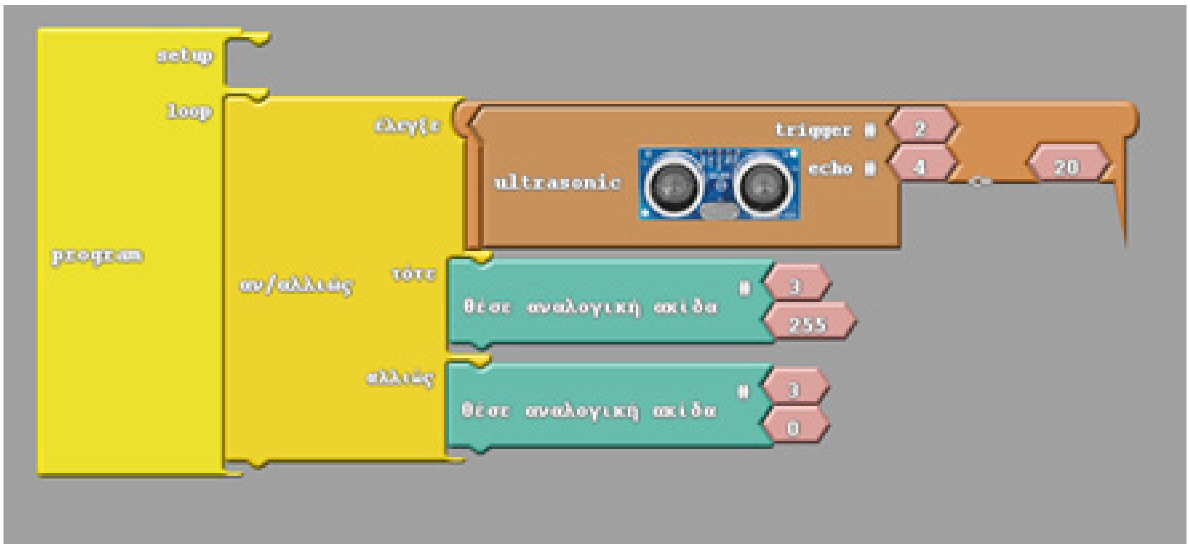
Echo goes to Arduino's PWM 4

GND goes to Arduino's GND

The buzzer must be connected to PWM 3, as shown in a previous lesson.

Activity 2: Programming the alarm device

Now the students are asked to program their device so that, as soon as something comes within the set range of the ultrasonic sensor (let's say within a distance of 20 cm), the buzzer sounds (when the "intruder" goes away the buzzer stops). The students must use the **<=** comparison operator from the orange **Tests** menu in the control condition of the **if/else** block. As seen in the screenshot of the code below, the rest is made easy with ArduBlock, since it uses a built-in **ultrasonic** function (found in the brown menu: **Generic Hardware**) which automatically performs all the calculations and only prompts the user for the numbers of the Trig and Echo pins.



The plain code is as follows:

loop [if (ultrasonic Trig 2 Echo 4) <= 20 *(the ultrasonic sensor connected to PWM 2 and 4 detects a distance < 20cm)*

then set analog pin 3 255 *(the buzzer connected to PWM 3 sounds the tone 255)*

else set analog pin 3 0] *(the buzzer connected to PWM 3 sounds the tone 0, i.e. no sound at all)*

Activity 3 (optional): Making a more realistic alarm system

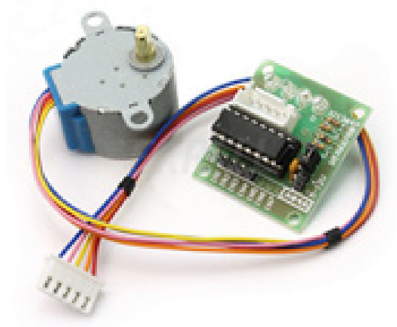
If the level of the students allows it, the teacher may suggest building a more sophisticated alarm model, with a built-in LED that indicates if the alarm is armed or not and making use of boolean variables (which will be described in the lessons that follow) that will enable arming and disarming the alarm. If the alarm is armed and something approaches the protected area, then the alarm sounds. However, as in real alarm systems, once the security system has been violated, the alarm should not cease sounding even if the intruder leaves the controlled perimeter. Instead, it must be disarmed using a remote control, for example.

Assessment

Student assessment is based on group work observation and how effectively they have managed to make a functioning anti-theft alarm system.

Controlling motors

Introduction: No device with electronic sensors is considered robotic if it doesn't contain at least one mechanical part whose motion is sensor-controlled, i.e., if it doesn't contain at least one sensor-activated motor. Arduino-compatible motors belong to three broad categories: servomotors (left photo below), which move with high precision from 0 to 180 degrees and backwards (that is, they do not make full rotations), DC motors (middle photo below), which are high-speed and low-strength motors that are used to move the wheels of toy-vehicles, and stepper motors (right photo below). The latter are more powerful motors whose full rotation is divided into a number of equal steps and thus they are more difficult to program. Servo motors are very suitable for controlling robotic arms, for example. That's why they are the first motors we should study. Activities with DC motors will also be presented in the next chapter.



Overview & Purpose: Connecting a servomotor to Arduino and programming it with ArduBlock will give students the opportunity to learn, how to use servomotors, variables within FOR-loops, variable value initialization and variables as counters.

Teaching Methods: Demonstration, Guided discovery, Trial and Error, Hands-On activities, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Connect a servomotor.
2) Program the servomotor to make it gratually turn clockwise and counter-clockwise.
3) Use variables in ArduBlock.
4) Use loops with a definite number of iterations.
5) Initialize a variable-counter outside a loop.
6) Increase and decrease a variable-counter inside a loop.
7) Use mathematical operators in ArduBlock.

Materials & Resources: Computers with Arduino IDE and ArduBlock installed. The Arduino boards, servomotors and jumper wires.

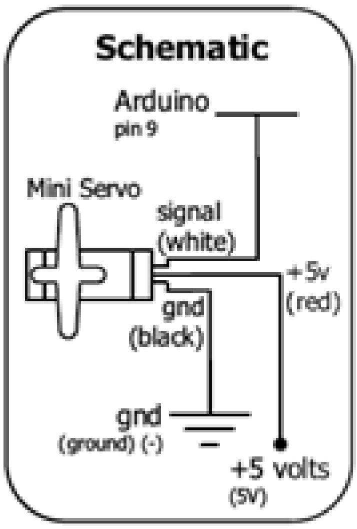
Activities

Activity 1: Connecting the servomotor

A servomotor has three wires attached to it and the color of the wires does matter! The red wire in the middle must always be connected to Arduino's 5V pin, while the black (or sometimes brown) wire must always be connected to Arduino's GND and the white (or sometimes yellow or orange) wire must be connected to a PWM pin, say PWM 9 (see the diagram below).

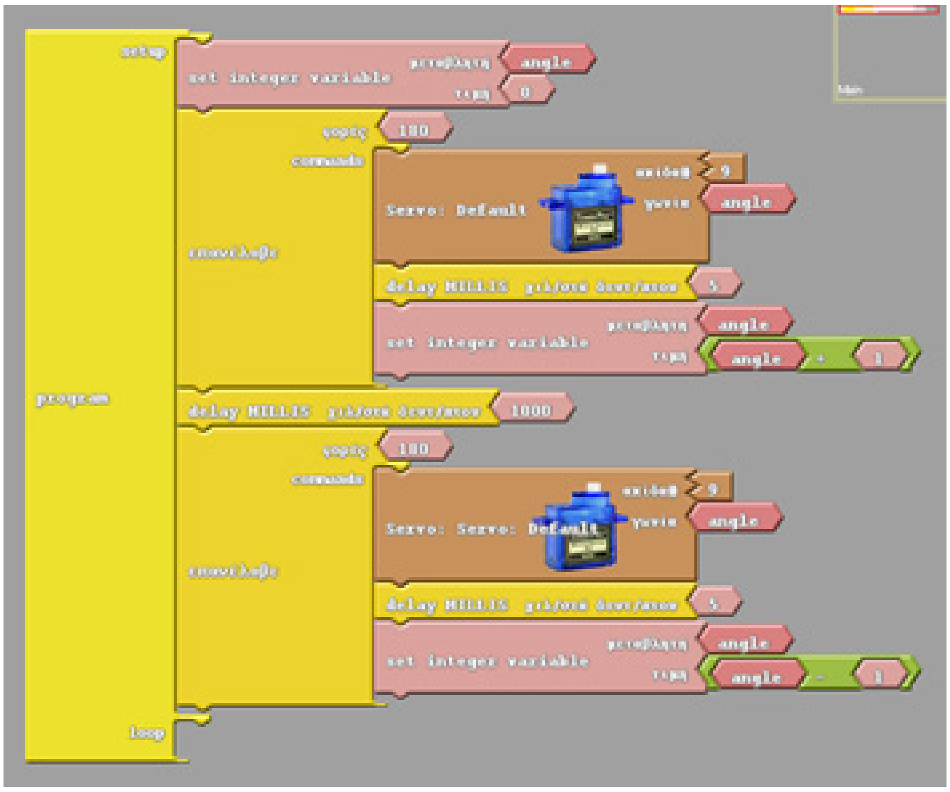
Activity 2: Programming the servomotor

Now, the students are asked to program the servomotor to move once from 0 to 180 degrees clockwise and then from 180 degrees back to 0 counter-clockwise. In order to make the program, the students must use the **setup** branch of the ArduBlock **program** block, since their commands will only be executed once, and connect a **repeat** block there from the yellow **Control** menu. The number of loop iterations must be set to 180 by connecting a pink rhomboid block from the pink **Variables / Constants** menu to the upper part of the repeat block.



The **Servo** block from the brown **Generic Hardware** menu must be connected to the lower part of the **repeat** block. The PWM pin, where the servomotor's white wire is connected, must be declared in the Servo block and an **integer variable name** must be connected from the pink **Variables / Constants** menu to hold the angle of the servomotor. This variable has to be previously declared, i.e. it must be declared outside and above the loop, using a suitable name, e.g. **angle**. A yellow delay command and the + arithmetic operator from the green **Math Operators** menu must also be included in the loop, in order to increase the variable value by one each time.

Finally, the part of the program that turns the servomotor counter-clockwise can be implemented with another repeat loop. This time, the value of the motor angle should gradually decrease from 180 to 0 degrees (see screenshot of the code below).



```
program
  setup
    set integer variable  [illegible] angle
                          [illegible] 0
  loop
    [illegible]  [illegible] 180
                 commands
      Servo: Default   [illegible] 9
                       [illegible] angle
      delay MILLIS [illegible] 5
      set integer variable  [illegible] angle
                            [illegible] angle + 1
    delay MILLIS [illegible] 1000
    [illegible]  [illegible] 180
                 commands
      Servo: Servo: Default   [illegible] 9
                              [illegible] angle
      delay MILLIS [illegible] 5
      set integer variable  [illegible] angle
                            [illegible] angle - 1
```

The plain code is as follows:

```
setup  [ set integer variable angle: 0
        repeat 180 times
                [ servo pin 9 angle
                delay 5 milliseconds
                set integer variable angle:  angle + 1 ]
        delay 1000 milliseconds
        repeat 180 times
                [ servo pin 9 angle
                delay 5 milliseconds
                set integer variable angle:  angle - 1 ]
       ]
```

Assessment

Students assessment is based on the effectiveness of their code in Activity 2.

Going further with ArduBlock

ArduBlock can be used to accomplish a large amount of Arduino projects, since the newest versions include built-in functions and libraries for a wide variety of electronics and Arduino compatible hardware. Here are some interesting ideas:

- Use a potentiometer to control the light intensity of a LED
- Output a sensor's input to the Serial Monitor
- Use various kinds of motors
- Use various kinds of displays

etc.

However, if the students' age and level of competence allows it, it might be more challenging to eventually introduce the students to the "real" Arduino programming environment in C, after they have completed a round of activities using ArduBlock. In this way, they will have the opportunity to benefit from the deluge of information, projects and code that is freely available thanks to the Arduino internet community, if they want to carry out more sophisticated Arduino projects in the future.

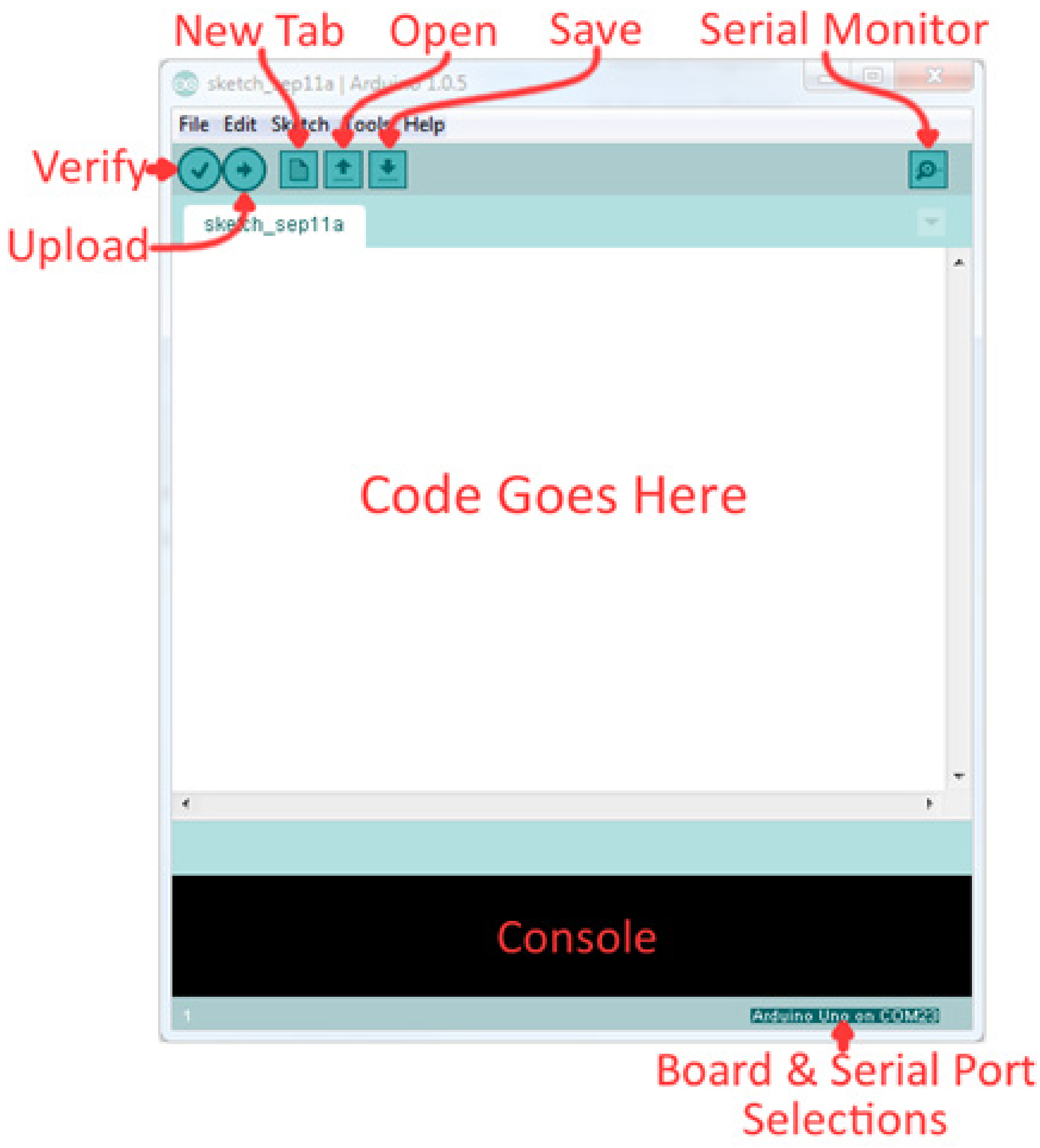
Bearing this in mind, the next chapter presents some introductory activites and projects using Arduino Sketches in C.

ADVANCED ROBOTICS PROGRAMMING

Potentiometer + LED: an introduction to the Arduino IDE

Introduction: This chapter presents an introduction to the Arduino Integrated Development Environment (IDE) and the language C. As mentioned at the end of the previous chapter, using this environment, the novice Arduino programmer can benefit from the enormous amount of projects and code that is already freely available thanks to the internet community, as well as exploit all Arduino features. In the lessons that follow, we present relatively simple Arduino activities that are different from the ones included in the ArduBlock chapter. However, they can also be implemented using ArduBlock - so that teachers and students have the opportunity to gradually learn how to use, not only the Arduino IDE but also the basic structures of the C language through experiential activities. The extent to which the lessons that follow can be implemented, depends mainly on the students' age and level of knowledge and the teachers' educational objectives.

A picture of the Arduino IDE explaining the environment icons is shown below (SparkFun, w.d.b).



Overview & Purpose: This project is an interesting example that is simple enough to introduce the Arduino IDE and Arduino Sketches. Students are offered the opportunity to exploit what they learned in the previous chapter and make a circuit that controls the light intensity of a LED with a 10 kOhm potentiometer. In parallel with the new environment, students will also learn what a potentiometer is and how to connect it and furthermore, they will experience the difference in range between Analog Input and Analog Output.

Teaching Methods: Demonstration, Brainstorming, Discussion, Guided Discovery, Hands-On activities, Trial and Error, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

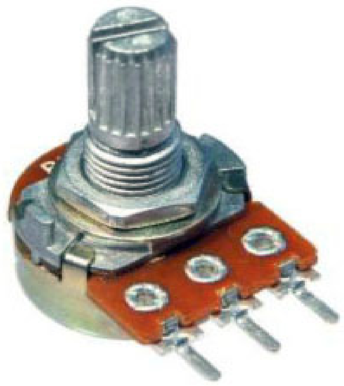
1) Describe the use of a potentiometer and how to connect it to the Arduino board.
2) Write simple C code in Arduino IDE and upload it to the Arduino board.
3) Describe what a C void function is and what is the difference between the **setup** and the **loop** Arduino structures.
4) Describe the use of **pinMode**, **analogWrite** and **analogRead** Arduino commands.
5) Know the difference between **digital** and **analog** values.
6) Know the difference in range between Arduino **analog input** and **analog output** values.
7) Use the seperators **() {} , ;** and arithmetic operators in C programming language.

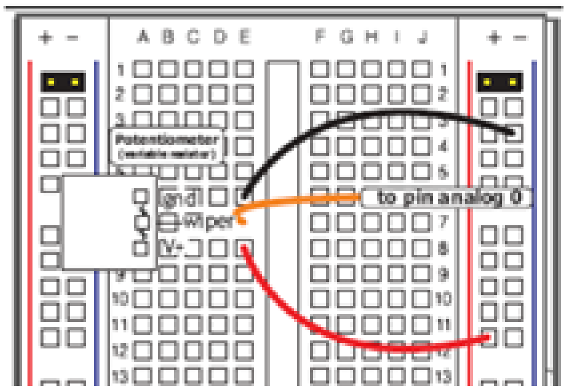
Materials & Resources: Computers with the Arduino IDE installed. The Arduino boards, breadboards, 10 kOhm potentiometers, LEDs, resistors and jumper wires.

Activities

Activity 1: Making the circuit

The students are given potentiometers (shown in the picture on the left below) and a discussion is initiated through the brainstorming technique on what they are used for. The teacher demonstrates how they must be connected: the left and right leads are connected to the board's 5V and GND pins, while the middle lead must be connected to any of the board's **ANALOG IN** pins (e.g. A0). The best way to accomplish this is by attaching the potentiometer directly on the breadboard, as shown in the image on the right below. Turning the potentiometer towards the lead where we have connected the 5V, its input value increases. So, if we want to turn the potentiometer clockwise in order to increase its input value, then the right lead (as seen from above) must be connected to the 5V pin and the left lead must be connected to a GND pin.





The students are also asked to connected a LED on Arduino's PWM pin 10 together with a 220 Ohm resistor, the same way it was shown in the activities of the previous chapter.

Activity 2: Programming with the Arduino IDE

The Arduino Sketch to be prepared by the students in order to program their connected LED to increase and decrease its light intensity as they turn the potentiometer is shown below:

```
void setup()
{ pinMode (10, OUTPUT); }
void loop()
{ analogWrite (10 , ( analogRead(A0) / 4 )); }
```

These commands instruct the Arduino microcontroller to do the following:

i) use PWM pin 10 (where the LED is connected) for **OUTPUT** (this command has to be executed only once in the beginning of the program and that is why it is included in the **setup** part of the code).

ii) Read an analog value from the analog pin A0 (where the potentiometer is connected), divide it by 4 and send the analog result (output) to pin 10. That is, give the appropriate current to the LED so that it emits light with intensity set through the potentiometer.

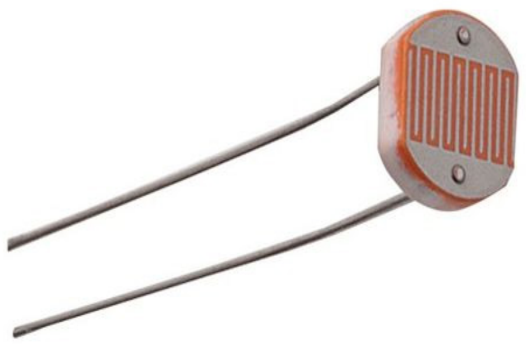
What the teacher must clarify to the students, is the difference between digital quantities, which take two distinct values only (**1**, representing a closed circuit, and **0**, representing an open circuit, or in Arduino IDE, **HIGH** and **LOW** respectively) and analog quantities. Usually, the range of analog values in electronics differs between analog input and analog output. In fact, Arduino IDE quantities that are used for analog output, can take values between **0 and 255** - and this is the range of values that may be passed to the **analogWrite** command. On the other hand, the **analogRead** command can get values from **0 to 1023**. This is why we often adopt the strategy of dividing an analog input value by 4 before passing it as analog output. A definitely more accurate technique is to use the Arduino built-in **map()** function, but, from a teacher's point of view, it may not be appropriate to use the **map()** function in the first Arduino IDE lesson!

Assessment

Student assessment is based on group work observation and the functionality of their code when uploaded to the Arduino board.

Detecting light and outputting to the Serial Monitor

Introduction: An easy way to detect ambient light intensity is through a photoresistor (also named photocell), seen in the image below. The resistance of a photoresistor decreases when light intensity increases and it is the electronic equipment normally used in devices that automatically switch on a building's lights when darkness falls!



Overview & Purpose: The activities below will not only allow students to learn what is a photocell and how it can be connected, but also how to output sensor measurements to the Serial Monitor, i.e. the computer screen that is connected to the Arduino board via USB.

Teaching Methods: Demonstration, Guided discovery, Hands-On activities, Trial and Error, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Describe what a photocell is and how it is connected to the Arduino.
2) Distinguish the various kinds of resistors.
3) Interpret photocell measurements to distinguish daylight from darkness and some colors.
4) Direct output to the Serial Monitor.
5) Make an automatic ambient light switch.

Materials & Resources: Computers with the Arduino IDE installed. The Arduino boards, breadboards, photoresistors, LEDs, resistors and jumper wires.

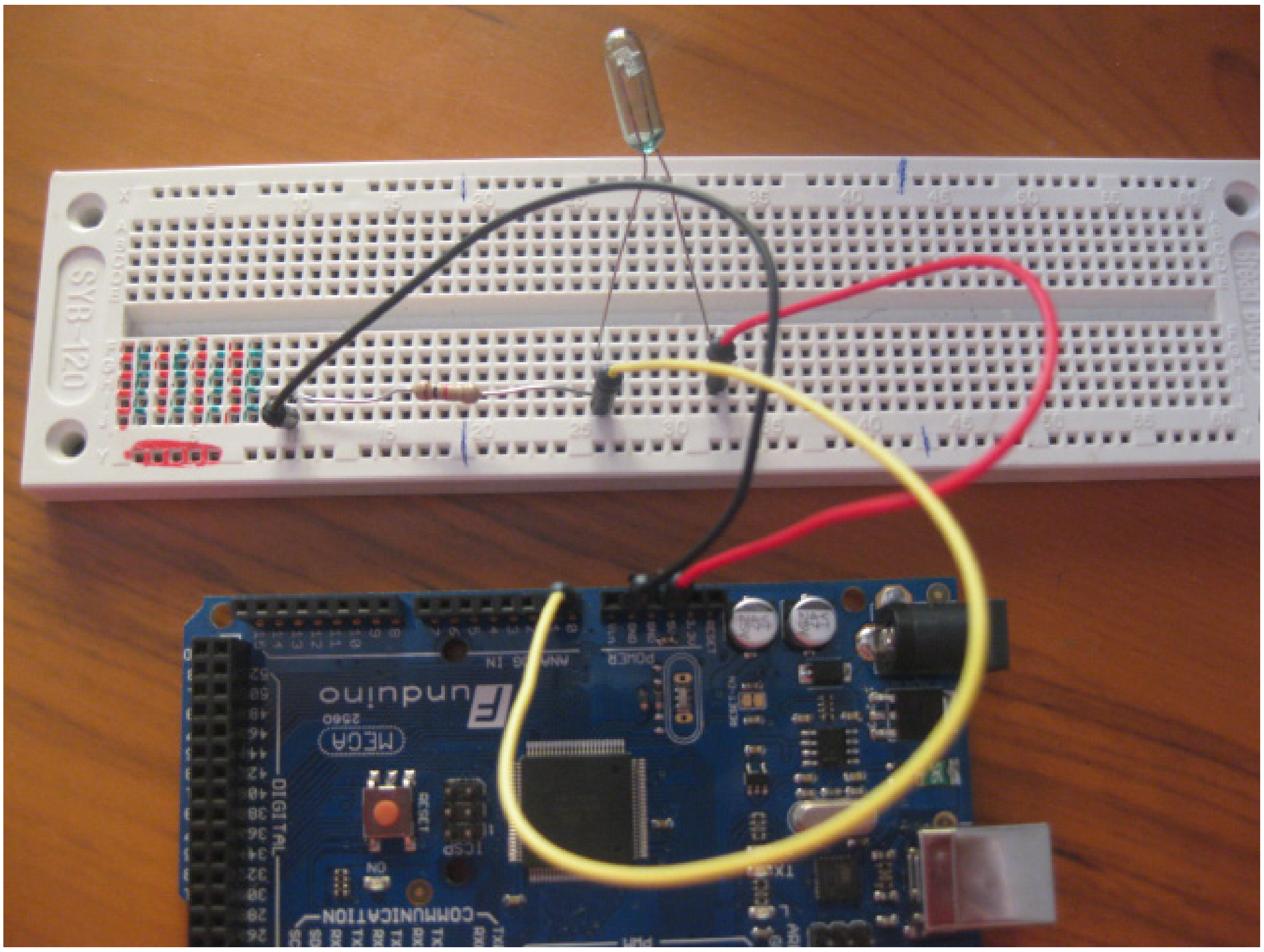
Activities

Activity 1: Making the circuit

Students are asked to fit the photocell on the breadboard. Then, they must connect one photocell lead to Arduino's 5V pin and the other photocell lead to a 1 kOhm resistor (the resistor colors are: brown, black and red). The other lead of the resistor must go to Arduino's GND, while the common line of the photoresistor and the resistor must also be connected to one of the board's ANALOG IN pins, say A0 (see photo on the next page).

Activity 2: Directing output to the Serial Monitor

In order to be able to see the photocell readings when the circuit is powered on, that is, the board is connected to the PC via USB, students must upload the following Arduino Sketch to the board:



```
void setup()
{ Serial.begin(9600); }
void loop()
{ Serial.println ( analogRead(A0) );
 delay(200); }
```

These commands instruct the Arduino microcontroller to do the following:

i) Initialise communication between the board and the PC monitor (9600 is the data transmission rate). This command has to be executed only once in the beginning of the program and that is why it is included in the **setup** part of the code.

ii) Send the values read from analog input A0 pin, where the photocell is connected, to the serial monitor every 200 milliseconds (so that the values scroll on the monitor at a readable rate).

As mentioned in the previous lesson, the range of values returned from the analog input pin varies from 0 to 1023. Testing the above program in various light conditions, the students will be able to see that the usual bright daylight values range from 200 to 300. When pointing a strong torch light directly to the photocell, the measurements rise up to 600. The photocell readings under an ordinary room lamp are usually below 100. Theoretically, the photocell might also be used to detect various colors, since it returns different values whan placed right in front of objects of different colors, made from the same material (e.g it returns the value 15 in front of a blue plastic cup and 25 in front of a red one (of the same material), due to different percentages that each color reflects.

Activity 3 (optional): Making an automatic light switching system

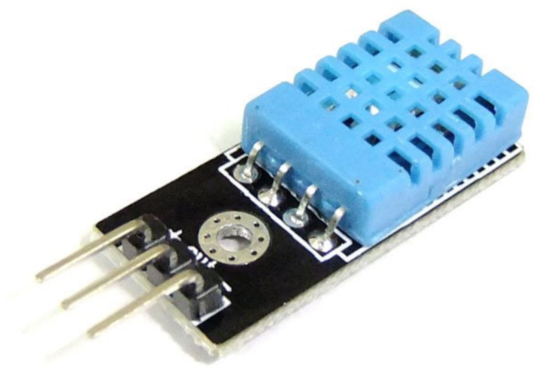
As further work, the students may be assigned to connect a LED to the circuit, as they have done in previous lessons. The LED must light when the photocell indication falls under the value 100, thus making an automatic light switching system! The LED light must switch off when it's "daylight" (a torch may be used for the tests) and ambient light intensity increases over 100. However, students must be careful with their construction. The photoresistor must be safely apart from the LED, because its measurements will be affected by the LED light and the system could be switching on and off continuously! In order to implement this activity, the teacher should present an introduction to the **if - else** clause in C language.

Assessment

Students assessment is based on worksheet correction and direct feedback during the activities. The teacher also observes the students' group work and assesses the execution of their programs on their robotic vehicle.

Measuring room humidity and temperature

Introduction: As mentioned in a previous lesson, the greatest advantage of using the Arduino platform is probably the fact that one can make use of a wide variety of electronic equipment and the corresponding C libraries that are freely offered either by the manufacturers of this equipment or by other developers around the world. The use of the DHT11 temperature-humidity sensor (shown below) is such an example. This sensor measures both room humidity and temperature and returns both of these values on its middle lead (for the technical specifications see D-Robotics, 2010). Using the provided DHT11 library, the Arduino programmer can easily separate the two values.



Overview & Purpose: In this lesson the students will learn how to install and use C-libraries (header files) in their code. They will also learn how to connect and get input from the DHT11 humidity-temperature sensor. Finally they will realise how important it is to comment their code.

Teaching Methods: Demonstration, Hands-On activities, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

1) Describe what a DHT11 humidity-temperature sensor measures and how it can be connected to Arduino.
2) Locate a developer's library for a specific electronic equipment and install it in the Arduino libraries.
3) Use a C-library (header-file).
4) Comment the code and describe why commenting is important.
5) Desribe the difference between the **print** and **println** commands in C-language.
6) Get input from the DHT11 humidity-temperature sensor and output it on the Serial Monitor.

Materials & Resources: Computers with the Arduino IDE installed. The Arduino boards, DHT11 sensors and jumper wires.

Activities

Activity 1: Making the circuit

Students are asked to connect the left and the right leads of the DHT11 sensor to Arduino's 5V and GND and the middle lead to a PWM pin, say PWM 2.

Activity 2: Installing the DHT11 library

The DHT11 library (as any other C-library) is nothing more than a C-header file (DHT11.h) that contains a number of ready-to-use functions in order to better exploit DHT11's hardware capabilities. It can be downloaded from numerous developer sites, such as http://dalxxdht11.blogspot.gr/2012/12/dht11-library-for-arduino-uno.html. The teacher must help the students locate a place to download the DHT11 library, download it, then make a folder named **DHT11** in the Arduino **libraries** folder and copy the DHT11.h file there.

In this way, the students will learn how to search for open-source software that other developers are kind enough to share with the community and how to install a new Arduino library.

Activity 3: Making the program

The Arduino Sketch to be prepared by the students must create in order to use the DHT11 library and get values from the humidity-temperature sensor is shown below:

```
#include <dht11.h>
dht11 DHT11;
void setup()
{ Serial.begin(9600); }
void loop()
{ int chk = DHT11.read(2);  // since the middle DHT11 sensor's lead is connected to PWM 2
  Serial.print("Humidity (%): ");
  Serial.println(DHT11.humidity);
  Serial.print("Temperature (in Celsius): ");
  Serial.println(DHT11.temperature);
  Serial.println();  // in order to leave a blank line in the screen between each sensor reading
  delay(2000); }
```

These commands instruct the Arduino microcontroller to do the following:

i) Use the **dht11.h** header file to exploit the **dht11** class and the **read** method to store the sensor values in the seperate **humidity** and **temperature** variables.

ii) Initialise communication between the board and the PC-monitor (with 9600 as the data transmission rate).

iii) Print the values of the **humidity** and **temperature** variables on the Serial Monitor every 2000 milliseconds (that is, 2 seconds).

iv) Use the comments separator **//** in order to insert comments and explain some parts of the code.

Note: It is expected that the students will not be able to thoroughly understand all the commands in the above code. The teacher should not put too much effort trying to explain every line of the code, but focus on the "easiest" parts. For example, s/he could explain the difference between **Serial.print()** and **Serial.println()** and stress the importance of commenting the difficult parts of the code.

Assessment

Student assessment is based on group work observation and how effectively they have managed to make their circuit work.

DC motors

Introduction: As also mentioned in the previous chapter, DC motors are the high-speed and low-strength motors that are usually used as wheels of model vehicles. That's why, if one wants to built any sort of moving robotic structure, one must know how to connect and program them.

Overview & Purpose: Connecting and programming DC motors. Using an H-bridge in order to control the rotation direction of a DC motor.

Teaching Methods: Demonstration, Guided Discovery, Hands-On activities, Group work, Collaborative learning

Duration: 3 hours

Objectives/targets

After completing the full set of activities described below, the students will be able to:

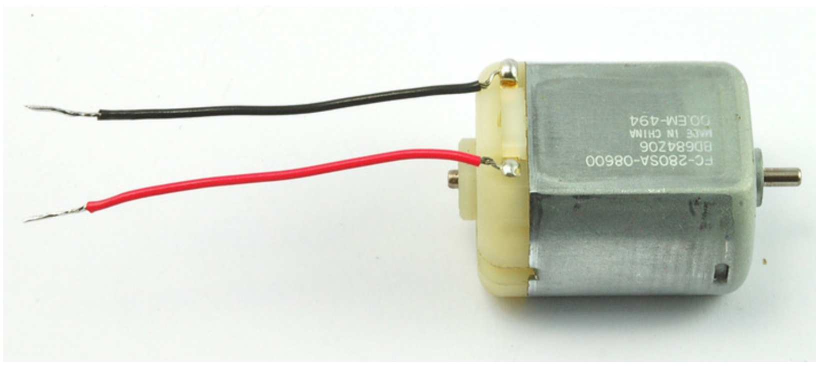
1) Describe how a DC motor works and how it can be connected to Arduino.
2) Describe how an H-Bridge circuit can be used to change the DC motor's rotation direction.
3) Make their own functions in C and call them in their program.
4) Describe what function parameters are.
5) Use the **int** and **boolean** data types to define variables.

Materials & Resources: Computers with the Arduino IDE installed. The Arduino boards, DC motors, H-Bridges, bread-boards and jumper wires.

Activities

Activity 1: Connecting the DC motor

As shown in the photo below, there are two wires coming out of the DC motor: a black one and a red one. By convention, the black wire is the cathode and so it must be connected to the Arduino GND, while the red one is the anode, so it can be connected to any PWM pin, say PWM 10, in order to send an analog value and set the motor rotation speed. However, if the two wires are connected the other way round, the only difference is that the motor rotates in the opposite direction.



Activity 2: Programming the DC motor

In order to program the DC motor, students can write a very simple program similar to the one they have written before for controlling the luminocity of a LED:

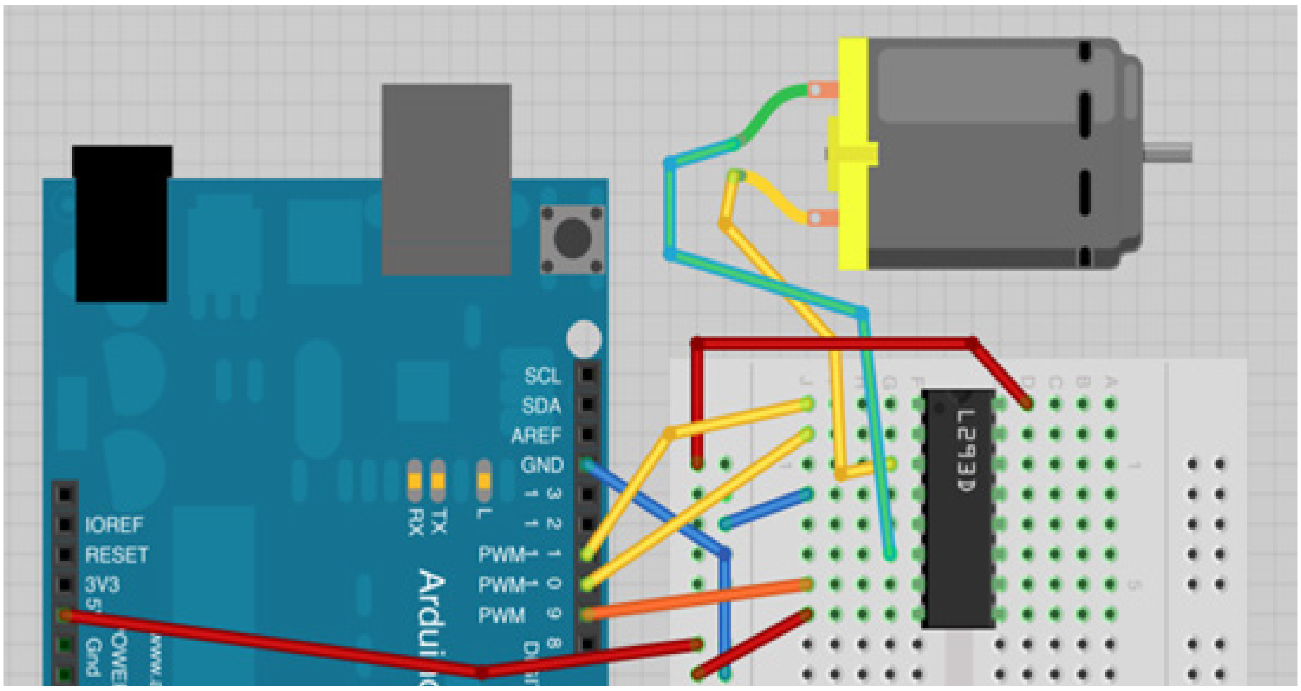
```
 void setup()
{ pinMode (10, OUTPUT); }
void loop()
{ analogWrite (10 , 200); }
```

In this case, the number 200 is a sample value between 0 and 255 used to set the motor rotation speed (bear in mind that very small values - e.g. less than 30 - may not have the capacity to move the rotor at all). Alternatively, a potentiometer can be used to set the motor rotation speed, as we have done in a previous lesson to control a LED. In order to better visualise the motor rotation, a small piece of colored tape may be attached to the motor axis like a flag.

However, if the student's level allows it, the teacher may pose the question: "What happens, though, if we want to change the motor rotation direction, e.g. if we want to make a model-car move backwards?" Of course it wouldn't be possible to change the motor wiring each time! Therefore, the answer to this question is to use an H-Bridge L293 integrated circuit (seen in the photo below), which does exactly that: it changes the current direction through programming.

Activity 3 (optional): Connecting the H-Bridge

Students must be instructed to reconnect the DC motor and the H-Bridge as shown in the diagram below (Monk, w.d.b)



Specifically, the DC motor wires are connected to the H-Bridge's 3rd and 6th lead (counting from the bottom left of the H-Bridge), while the H-Bridge's 1st, 2nd and 7th lead are connected to the PWM pins 11, 10 and 9 respectively. The 4th H-Bridge's lead goes to GND while the 8th bottom and the 1st top lead of the H-Bridge are both connected to the 5V output through a breadboard line. The rest of the H-Bridge's leads remain unconnected.

Activity 4 (optional): Programming the H-Bridge

The Arduino Sketch to be prepared by the students in order to program the DC motor and change rotation direction through the H-Bridge is shown below:

```
void setup()
{ pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT); }

void loop()
{ setMotor(100, 1);
 delay(1000);
 setMotor(100, 0);
 delay(1000); }

void setMotor(int speed, boolean direction)
{ analogWrite(11, speed);
  digitalWrite(10, !direction);
  digitalWrite(9, direction); }
```

The PWN 11 pin is used as analog output to pass the motor speed while both the PWM 10 and 9 pins are used as digital output to set the motor rotation direction. The values of pins 10 and 9 must always be reverse (that is, if one is 1 (HIGH) the other must be 0 (LOW) and vice-versa).

setMotor is a user-specified function that sets the motor speed and rotation through its two parameters (the integer **speed** and the boolean **direction**).

In the **loop** section of our program, the **setMotor** function is called twice, since the motor turns continuously at speed 100, towards one direction for 1sec (determined by the **delay** command) and towards the other direction for another sec.

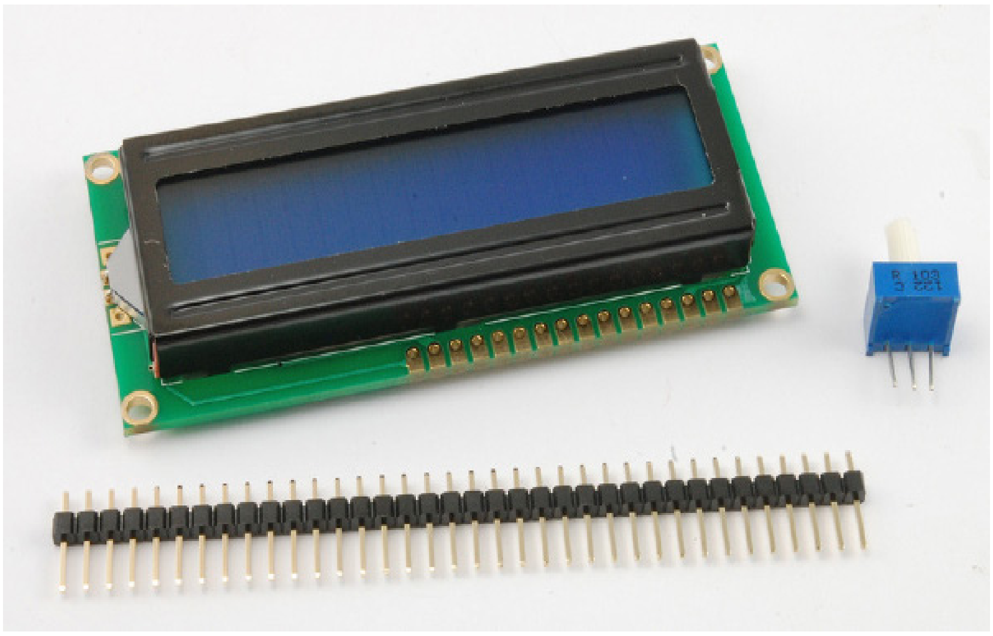
Using the above code as an example, the teacher must explain how we create our own functions in C, what are function parameters, how they are declared in the function code and how parameter values are passed when a function is called.

Assessment

Student assessment is based on group work observation and how effectively they have managed to make their DC motor rotate in both directions!

Directing output to an LCD display

Introduction: Most electronic devices have some sort of an output screen. In order for the students to get an idea of how such a thing works, it is most suitable to use a **Liquid Christal Display (LCD display),** e.g a display with 16x2 characters like the one shown below (Monk, w.d.a). The pins of the display come seperately and so they have to be soldered. Furthermore, a 10 kOhm variable resistor (shown on the right below) or the 10 kOhm potentiometer used in the first lesson of this chapter is needed in order to adjust the contrast of the display.



Overview & Purpose: The students will learn how the displays of various electronic devices (e.g. the scrolling displays in public transportation vehicles) work.

Teaching Methods: Demonstration, Guided discovery, Hands-On activities, Trial and Error, Group work, Collaborative learning

Duration: 2 hours

Objectives/targets

After completing the activities below, the students will be able to:

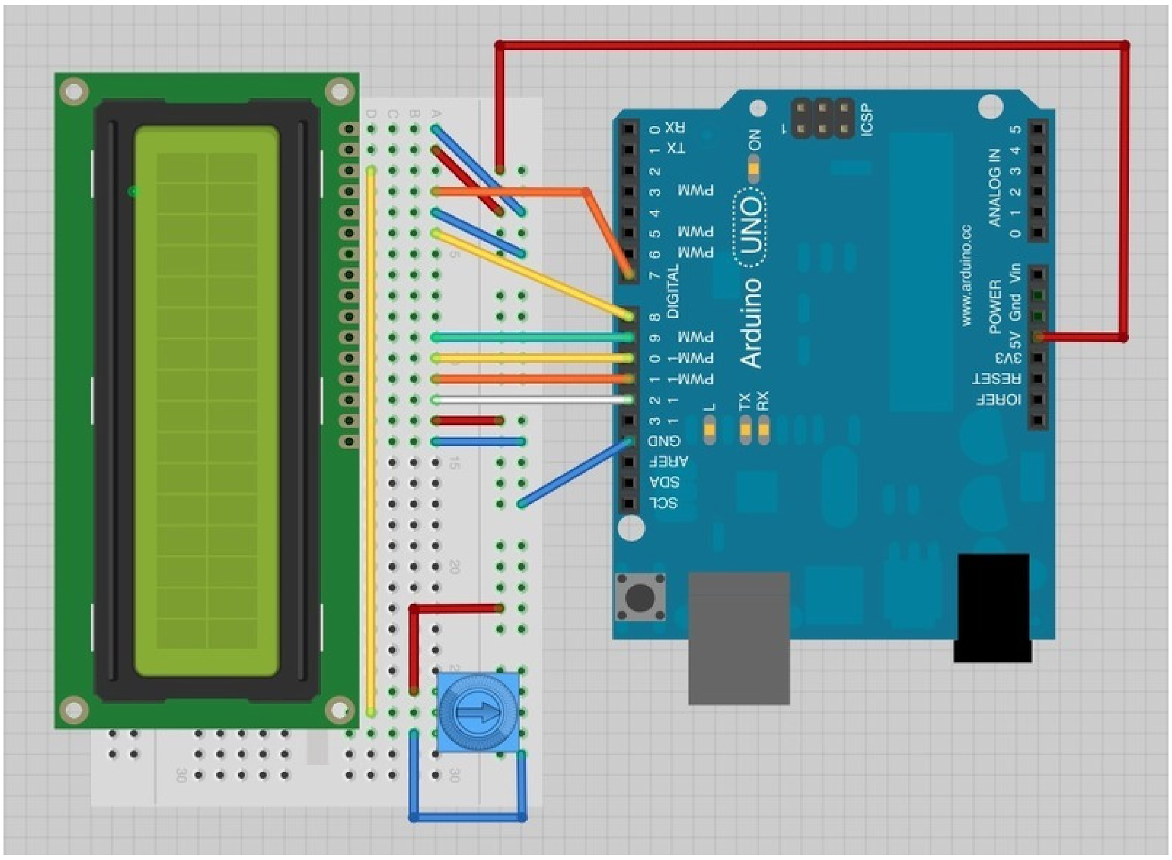
1) Describe how an LCD display works and how to connect it to Arduino.
2) Program the LCD circuit to display various messages.
3) Display scrolling messages.
4) Control the contrast of the LCD display with a potentiometer.

Materials & Resources: Computers with the Arduino IDE installed. The Arduino boards, breadboards, 10 kOhm potentiometers, LCD displays and jumper wires.

Activities

Activity 1: Connecting the LCD display

The LCD display is connected to Arduino as shown in the next diagram (Monk, w.d.a). In fact, pins 1, 5 and 16 of the LCD display are connected to Arduino's GND (numbering starts from the top, as seen in the diagram). Pins 2 and 15 are connected to Arduino's 5V pin. Pin 3 must be connected to the potentiometer's middle lead. Pins 4, 6, 11, 12, 13, 14 go to Arduino's PWM 7, 8, 9, 10, 11 and 12 respectively, while pins 7-10 remain unconnected.



Activity 2: Programming output on the LCD display

In order to program the LCD display, the **LiquidCrystal** library must be first installed in the **libraries** folder of the Arduino installation. An introductory Arduino Sketch to help students familiarize themselves with the various **LiquidCrystal** methods can be as follows:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);  // the Arduino PWM pins where the LCD is connected
void setup()
{ lcd.begin(16,2);          // the display will have 16 columns and 2 rows
 lcd.clear();               // clear the screen
 lcd.setCursor(0,0);        // set cursor to column 0, row 0 (first row)
 lcd.print("I love");       // this text will appear at the left-hand side of the first row
 lcd.setCursor(0,1);        // set cursor to column 0, row 1 (second row)
 lcd.print("Arduino!");     // this text will appear at the left-hand side of the second row
 }
void loop() {}
```

In this way, the message

I love

Arduino

appears constantly (if it doesn't appear, the contrast must be adjusted with the help of the potentiometer). If we want the same message to scroll, we have to include the following code in the **loop** function:

```
void loop()
 { lcd.setCursor(16,1);     // set the cursor outside the display count
   lcd.autoscroll();        // set the display to automatically scroll
   lcd.print(" ");          // print empty character (necessary to start scrolling)
   delay(300);              // this determines the scrolling speed
 }
```

Assessment

Student assessment is based on group work observation, how effectively they have managed to connect the LCD display, and if their message appears on the screen.

A DIY device

Introduction: This is the final Arduino project of the series of robotics and automation activities described in this issue. It combines previous knowledge obtained from the projects on the DHT11 humidity and temperature sensor and the LCD display. The students are asked to work on their own and create a battery-operated electronic thermometer (like the ones sold in the market!) that displays room humidity and temperature on an LCD screen. Students should be highly motivated bearing in mind that they are working on their first real DIY (Do It Yourself) device!

Overview & Purpose: Making a complete functioning battery-operated electronic device. Becoming creative by embedding and exploiting previous knowledge.

Teaching Methods: Guided Discovery, Hands-on activities, Trial and Error, Group work, Collaborative learning

Duration: 3 hours

Objectives/targets

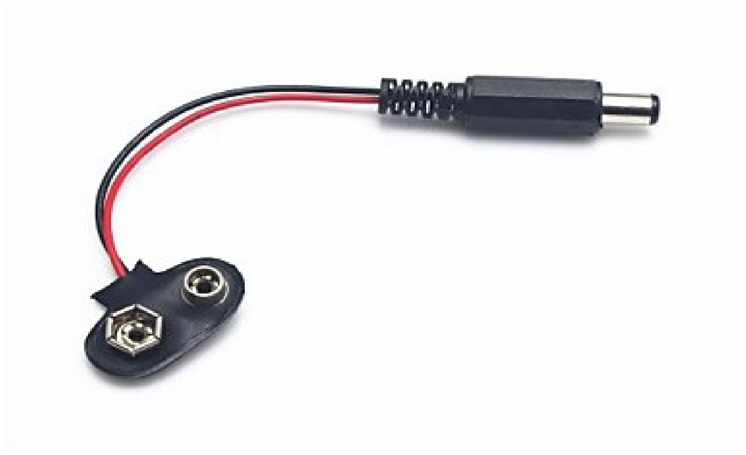
At the end of this lesson, the students will be able to:

1) Apply previous knowledge into new creations of their own.
2) Make a battery operated electronic device based on the Arduino board.
3) Combine pieces of code from other applications in their own new application.
4) Use the **float** converter and determine the number of decimal digits to be printed.

Materials & Resources: Computers with the Arduino IDE installed. The Arduino boards, breadboards, LCD displays, 10 kOhm potentiometers, DHT11 sensors, jumper wires, 9V battery connector cables for Arduino, 9V batteries.

Activity

Students are shown the new battery and connector equipment (see photo below) for making battery-operated devices. They are asked to apply what they have learned in previous lessons about connecting and programming the DHT11 sensor and the LCD display in order to create a battery-operated electronic thermometer that displays room humidity and temperature on the LCD display. Students are not expected to encounter problems with connecting the above equipment, since the procedure is exactly as described in the previous lessons.



The code students must create combining the Arduino Sketches they have used in previous projects is the following:

```
#include <LiquidCrystal.h>
#include <dht11.h>
dht11 DHT11;
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

```
void setup()
{ lcd.begin(16,2);          // initialize the LCD display
 lcd.clear();               // clear the screen
 }

void loop()
{ int chk = DHT11.read(2);                  // DHT11 is connected to PWM 2
 lcd.setCursor(0,0);
 lcd.print("Humidity:  ");
 lcd.print((float)DHT11.humidity, 1);       // the number is converted to float with one decimal digit
 lcd.setCursor(0,1);
 lcd.print("Temperature:");
 lcd.print((float)DHT11.temperature, 1); // again float with one decimal digit
 delay(300);                                // determines the display refresh rate
}
```

The students must think on their own and come to the conclusion that temperature and humidity values must be printed inside the **loop** part of the code, since their values change with time. The teacher must only introduce the meaning of the **float** command and explain how to determine the number of decimal digits to be printed.

After uploading their sketch on the board and testing that it works properly, the students can **first disconnect the board from the PC and then connect it to a 9V battery via the provided connector cable**. Their program should work again just fine and their first DIY electronic device will be ready!

Assessment

Student assessment is based on group work observation and how effectively they have managed to make the connections, program their circuits and make their electronic device function!

Going further with Arduino

As already mentioned before, there is no limit to the types of projects that students can do with the Arduino platform. There is a wide variety of books which describe Arduino activities that address different levels of expertise. Indicatively, the following should be mentioned for further reading (in chronological order): Banzi (2008), Monk (2010, 2011a, 2011b), Smith (2011), Margolis (2012), McRoberts (2013). Numerous sites (such as Adafruit (2016), Instructables (2016) and Sparkfun (w.d.c)) are also important, as they provide ideas together with ready-made code for several Arduino projects.

Furthermore, before closing this issue, we should also mention the alternative of exploiting other similar SoC (System on Chip) platforms, such as Raspberry Pi (RaspberryPi, w.d.), Banana Pi (BananaPi, 2014) and BeagleBone (Beagleboard, w.d.). Today's technology offers many low-cost possibilities to be used on all levels of education and young people become familiar with new technology amazingly quickly. So, all that is really needed is innovation-loving and inspiring teachers...

Acknowledgements

I would like to thank the Head Teacher of the Experimental Junior High School of the University of Macedonia Dr. Eleni Mouzoura, not only for processing this issue, but mainly for the many years of excellent collaboration and continuous support to all the innovative ideas and pioneering robotics projects I have brought into life with my students while running the Robotics Club at our School. I would also like to thank the "Our School - My Future, ESL project" Project Manager Dr. Maria Machairidou and the Member of the Coordinating Team Mr. Efthimios Mavrogeorgiadis for editing and proof-reading this issue and for designing and coordinating this exquisit project that promoted the exchange of knowledge and ideas on Robotics and Educational Innovation between teachers and students from the participating European schools. Finally, I would like to thank my colleague Mrs. Styliani Koukourikou for representing our robotics team at the Erasmus+ meeting in Romania and for all the years of our excellent cooperation while teaching Informatics at the Experimental Junior High School of the University of Macedonia.

As I was bulding my robotics experience in education over the past few years, many people played an important role and I always feel thankful for that. Specifically, I am indebted to Prof. Nikolaos Fachantidis, the person that introduced me to Educational Robotics and lent me the first equipment to start experimenting, Prof. Vasilios Dagdilelis and Prof. Dimitrios Psyllos, who lent further equipment to our team, and the Mayor of Neapoli-Sykies Mr. Simos Daniilidis for offering new EV3 equipment to our Robotics Club and supporting all our activities.

My family is the "unknown hero" behind this robotics adventure, since they have put up with me during my endless hours of extra work at school and at home in front of the computer, hearing the permanent promise "It's only until the end of the project". However, as soon as one project finished, another project started...

Last but not least, I would like to thank all of my students who attended the Robotics Club and supported its activities. Their young spirit and enthousiasm was the source of my inspiration and the biggest reward for all my efforts. Special mention is due to Pavlos Theodosiadis, whose devotion, hard work and high talent in electronics and construction have helped our Club materialize not only the Arduino activities described in this issue, but also most of our award-winning projects.

Dr. Aikaterini Chatzifoteinou

References

Adafruit. (2016). Learn Arduino. Retrieved from https://learn.adafruit.com/category/learn-arduino.

Arduino (2016). Retrieved from http://www.arduino.cc

Bagnall, B. (2014). Maximum LEGO EV3 Programming. Variantpress.

Banana Pi. (2014). Banana Pi - A High-end Single-Board Computer. Retrieved from http://www.bananapi.org/p/product.html.

Banzi, M. (2008). Getting Started with Arduino. Make: Projects.

Beagleboard. (w.d.). BeagleBone. Retrieved from http://beagleboard.org/bone.

Benedettelli, D. (2013). The LEGO MINDSTORMS EV3 Laboratory: Build, Program, and Experiment with Five Wicked Cool Robots! No starch press.

Bratzel, B. (2014). STEM by Design: Teaching with LEGO Mindstorms EV3. Paperback

Citilab (2015). S4A. Retrieved from http://s4a.cat/.

Dagdilelis, V., Sartatzemi, M., & Kagani, K. (2005). Teaching (with) Robots in Secondary Schools: Some new and not-so-new Pedagogical problems. Fifth IEEE International Conference. ICALT 2005. Advanced Learning Technologies.

Datteri, E., Zecca, L., Laudisa, F., & Castiglioni, M. (2012). Explaining robotic behaviors: a case study on science education. In D. Alimisis & M. Moro (eds.), Proceedings of 3rd International Workshop Teaching Robotics, Teaching with Robotics Integrating Robotics in School Curriculum (pp. 134-143) Trento, Italy. Retrieved from http://www.terecop.eu/TRTWR2012/trtwr2012_submission_20.pdf.

Demetriadis, S., Atmatzidou, S., & Sapounides, T. (2012). The AUTh Framework for Research in Educational Robotics: Collaboration Scripts, Metacognitive Skills, Tangible Interfaces and the CPPC+ Model. 3rd International Workshop Teaching Robotics, Teaching with Robotics Integrating Robotics in School Curriculum, Riva del Garda, Trento, Italy, 20 April 2012. (pp.196-197).

Denis, B. & Hubert, S. (2001). Collaborative learning in an educational robotics environment. Computers in Human Behavior, 17, 465–480.

D-Robotics (2010). DHT11 Humidity & Temperature Sensor. Retrieved from http://www.micro4you.com/files/sensor/DHT11.pdf.

Erwin, B., Cyr, M., & Rogers, C. (2000). LEGO engineer and RoboLab: Teaching engineering with LabVIEW from kindergarten to graduate school. International Journal of Engineering Education, 16(3), 181-192.

Francis, K., Poscente, M., Friesen, S. & Davis, B. (2014). EV3 Robots Introduction to Programming. University of Calgary. Retrieved from http://www.ucalgary.ca/IOSTEM/teachers/ev3-robots-introduction-programming.

Garber, G. (2013). Instant LEGO MINDSTORMS EV3. PACKT Publishing.

Gibb, A.M. (2010). New Media Art, Design, And The Arduino Microcontroller: A Malleable Pool. Master's Thesis. Pratt Institute.

Gillig, H.S. (2014). miniBloq. Retrieved from http://blog.minibloq.org/2014/04/minibloq-redbot-opencv-free-tutorials.html.

Goff, R. M., & Vernon, M. R. (2001). Using LEGO RCX bricks as the platform for interdisciplinary design projects. In Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition.

Griffin, T. (2014). Art of LEGO MINDSTORMS EV3 Programming. No starch press.

Hacker, L. (2003). Robotics in education: ROBOLAB and robotic technology as tools for learning science and engineering. Unpublished Senior Thesis Department of Child Development, Tufts University. Retrieved from http://ase.tufts.edu/roboticsacademy/Theses/LauraHacker03.pdf.

Higashi, R. & Shoop, R. (2014). Introduction to Programming LEGO MINDSTORMS EV3 Teacher's Guide. Carnegie Mellon Robotics Academy. Retrieved from http://www.education.rec.ri.cmu.edu/content/lego/ev3/files/EV3%20teachers%20guideWEB.pdf

Huang, B. (2015). Open-source Hardware – Microcontrollers and Physics Education – Integrating DIY Sensors and Data Acquisition with Arduino. 122nd ASEE Annual Conference & Exposition, Seattle, WA.

Igoe, T. (2009). Making things talk. O'Reilly Media Inc.

Instructables. (2016). A beginner's guide to Arduino. Retrieved from http://www.instructables.com/id/A-Beginners-Guide-to-Arduino/?ALLSTEPS.

Isela, M., & Mota, G. (2007). Work in progress - using Lego mindstorms and Robolab as a mean to lowering dropout and failure rate in programming course. In Proceedings of the 37th annual frontiers in education conference - global engineering: knowledge without borders, opportunities without passports, 2007 FIE '07.

Isogawa, Y. (2014). LEGO MINDSTORMS EV3 Idea Book. No starch press.

Jonassen, D. H. (2000). Computers as mindtools for schools. NJ: Prentice Hall.

Karch, M. (2014). Build and Program Your Own LEGO Mindstorms EV3 Robots. Que publishing.

Kee, D. (2015). Classroom Activities for the Busy Teacher: EV3. Retrieved from http://www.damienkee.com/classroom-activities-ev3/

Kushner, D. (2011). The Making of Arduino. Retrieved from http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino

LEGO (2015). LEGO education. Inspirational Hardware. Retrieved from https://education.lego.com/nl-nl/lesi/middle-school/mindstorms-education-ev3/all-about-ev3/hardware

LEGO (2016a). The MINDSTORMS Community. 4 wheel line follower profile. Retrieved from http://www.us.lego.com/en-gb/mindstorms/community/robot?projectid=757a9f25-8729-4ad0-aad7-5540111ebcf1

LEGO (2016b). The MINDSTORMS Community. 3D Building Instructions. Retrieved from http://www.us.lego.com/en-us/mindstorms/community/bi/

Margolis, M. (2012). Arduino cookbook. O' Reilly Media Inc.

Martin, F., Mikhak, B., Resnick, M. Silverman, B. & Berg, R. (1987). Evolution of a Construction Kit for Magical Machines, MIT Media Laboratory. Retrieved from http://www.cs.uml.edu/~fredm/papers/magical-machines.pdf

McRoberts, M. (2013). Beginning Arduino. Apress: Technology in Action.

Mikropoulos, T. & Bellou, I. (2013). Educational Robotics as Mindtools, Themes in Science & Technology Education, 6(1), p. 5-14.

Modkit (2016). Modkit Micro. Retrieved from http://www.modkit.com/micro.

Monk, S. (2010). 30 Arduino Projects for the Evil Genius. McGraw-Hill/TAB Electronics.

Monk, S. (2011a). Arduino + Android Projects for the Evil Genius: Control Arduino with Your Smartphone or Tablet. McGraw-Hill Education TAB.

Monk, S. (2011b). Programming Arduino Getting Started with Sketches. McGraw-Hill Education TAB.

Monk, S. (w.d.a). adafruit - Arduino Lesson 11: LCD Displays – Part 1. Retrieved from https://learn.adafruit.com/adafruit-arduino-lesson-11-lcd-displays-1/breadboard-layout.

Monk, S. (w.d.b). adafruit - Arduino Lesson 15: DC Motor Reversing – Breadboard Layout. Retrieved from https://learn.adafruit.com/adafruit-arduino-lesson-15-dc-motor-reversing/breadboard-layout.

Nuget, G., Barker, B., Grandgenett, N., & Adamchuk, V. (2009). The Use of digital manipulatives in K-12: robotics, GPS/GIS and programming. In C. Atman (ed.). Proceedings of the 39th IEEE international conference on Frontiers in education conference (pp. 302-307) San Antonio Texas: IEEE Press.

Papanikolaou, K., & Frangou, S. (2009). Robotics as Learning Tool. In D. Alimisis (ed.), Teacher education on Robotics-enhanced constructivist pedagogical models (pp. 103-137), Athens: School of Pedagogical and Technological Education (ASPETE). Retrieved from http://dide.ilei.sch.gr/keplinet/education/docs/book_TeacherEducationOnRobotics-ASPETE.pdf.

Papert, S. (1980): Mindstorms: Children, Computers and Powerful Ideas. Basic Books.

Papert, S. & Harel, I. (1991). Situating Constructionism. Constructionism, Ablex Publishing Corporation: 193-206. Retrieved from http://www.papert.org/articles/SituatingConstructionism.html.

Parker, D. (2014). nxtprograms.com. Fun Projects for your LEGO® MINDSTORMS® NXT! Retrieved from http://nxtprograms.com/

Raspberry Pi. (w.d.). Teach, Learn and Make with Raspberry Pi. Retrieved from https://www.raspberrypi.org/.

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. In Proceedings of the Interaction Design and Children conference. Boulder, CO.

Ringwood, J. V., Monaghan, K., & Maloco, J. (2005). Teaching engineering design through Lego Mindstorms. European Journal of Engineering Education, 30(1), 91-104.

Robomatter (2014). ROBOTC - a C Programming Language for Robotics. Retrieved from http://www.robotc.net/

Rollins, M. (2014). Beginning LEGO MINDSTORMS EV3. Apress.

Smith, A. (2011). Introduction to Arduino – A piece of cake!. Retrieved from http://www.introtoarduino.com/downloads/IntroArduinoBook.pdf.

SparkFun (w.d.a). Alternative Arduino Interfaces. Retrieved from https://learn.sparkfun.com/tutorials/alternative-arduino-interfaces/all.

SparkFun (w.d.b). RedBoard Hookup Guide. Retrieved from https://learn.sparkfun.com/tutorials/redboard-hookup-guide.

SparkFun (w.d.c). MakerScience and Arduino. Retrieved from https://learn.sparkfun.com/resources/72.

Sullivan, F. R. (2008). Robotics and science literacy: thinking skills, science process skills and systems understanding. Journal of Research in Science Teaching, 45(3), 373–394.

Staszowski, K. J., & Bers, M. (2005). The effects of peer interactions on the development of technological fluency in an early-childhood, robotic learning environment. In Proceedings of the 2005 American Society of Engineering Education Annual Conference & Exposition.

Taweili, D.L. & Qichen, H. (2011). ArduBlock - a Block Programming Language for Arduino. Retrieved from https://github.com/taweili/ardublock.

Tewari, D. (2014). Programming an Arduino using BlocklyDuino. Retrieved from https://delog.wordpress.com/2014/06/10/programming-an-arduino-using-blocklyduino/.

Valk, L. (2013a). Robot Square. LEGO MINDSTORMS EV3 Education 45544 Instructions. Retrieved from http://robotsquare.com/2013/10/01/education-ev3-45544-instruction/

Valk, L. (2013b). Robot Square. LEGO MINDSTORMS EV3 Education Expansion Set 45560 Instructions. Retrieved from http://robotsquare.com/2013/10/01/lego-mindstorms-ev3-education-expansion-set-45560-instructions/

Valk, L. (2014). LEGO MINDSTORMS EV3 Discovery Book. No starch press.

Valk, L. (2015). Robot Square. EV3 Building Tutorials. Retrieved from http://robotsquare.com/category/tutorials/ev3-building-tutorials/

Wang, E. (2001). Teaching freshmen design, creativity and programming with LEGOs and LABVIEW. In Proceedings of the ASEE/IEEE Frontiers in Education Conference.

Watters, A. (2015). Lego Mindstorms: A History of Educational Robots. Retrieved from http://hackeducation.com/2015/04/10/mindstorms/.

ISBN 978-960-99808-2-1

Erasmus+

ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ IKY