What European teachers can learn from each other

SCIENCE ON STAGE GERMANY

THE EUROPEAN NETWORK FOR SCIENCE TEACHERS

Coding in STEM Education

English

<Imprint>

<Published by>
Science on Stage Deutschland e.V.
Am Borsigturm 15
13507 Berlin, Germany

<Main Coordinator>
↳ **Dr Jörg Gutschank**, Leibniz Gymnasium | Dortmund International School, Dortmund, Germany
Chair Science on Stage Deutschland e.V.

<Coordinators>
↳ **Sebastian Funk**, Villa Wewersbusch, Velbert-Langenberg, Germany, Board Science on Stage Deutschland e.V.
↳ **Jean-Luc Richter**, Lycée Jean-Baptiste Schwilgué, Sélestat, France, Vice Chair Science on Stage France
↳ **Bernard Schriek** (ret.), Marien-Gymnasium, Werl, Germany

<Overall Coordination and Editing>
↳ **Daniela Neumann**, Project manager
Science on Stage Deutschland e.V.
↳ **Stefanie Schlunk**, Executive manager
Science on Stage Deutschland e.V.
↳ **Johanna Schulze**, Deputy executive manager
Science on Stage Deutschland e.V.

<Revision and Translation>
Translation-Probst AG

<Design>
WEBERSUPIRAN.berlin

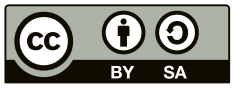
<Illustration>
Rupert Tacke, Tricom Kommunikation und Verlag GmbH

<Credits>
The authors have checked all aspects of copyright for the images and texts used in this publication to the best of their knowledge.

<Supported by>
SAP SE

<Please order from>
www.science-on-stage.de
info@science-on-stage.de

<ISBN PDF>
978-3-942524-58-2



Second edition published in 2019

<Content>

<Greeting>

I am very pleased to lend my support to this brochure regarding education in Science, Technology, Engineering and Maths – the important STEM subjects – and, more broadly, the project *Coding in STEM Education.*

The STEM disciplines are crucial in building a competitive, resilient Europe for the future. Building knowledge and excellence here is a key part of our ambition to build a true European Education Area by 2025. And yet, we are facing a skills gap. We need to do more to promote STEM subjects in Europe. In particular, teaching STEM subjects must be an attractive career choice, and we need more role models, especially women.

This is not only about economic growth and development. STEM education needs to be inclusive, enabling students of diverse abilities and backgrounds to engage and make the most of their talents. To build the future, young people need the right skills and attitudes – and STEM proficiency needs to be at the heart of these.

Through science we can learn so much: about our past, and our present, enabling us to shape a better, more knowledgeable, future. So that future generations can live in a society with a greater awareness of the world we live in.

I want to thank Science on Stage Deutschland e.V., the European Network for Science Teachers, which is very much the brains behind this operation, and SAP SE, for supporting coding projects.

Teachers from seven European countries helped develop concrete examples and practical advice on how to acquire coding skills. This shows what is possible when we get together, how much we have in common and how much we have to learn from each other.

Projects like *Coding in STEM Education* have a key role to play in promoting STEM subjects in schools, helping Europe's youth acquire vital competences they need to succeed in life. I commend all those involved and hope your example inspires others.

Tibor Navracsics
European Commissioner for Education, Culture, Youth and Sport

<Greeting>

According to a forecast of the World Economic Forum, around 65 per cent of children who start primary school today will work in professions that do not even exist yet. Nevertheless, it is clear that these children will find their feet more easily if they have certain technical skills; the digitalisation of our economy simply cannot be stopped. Therefore, in addition to reading, writing and math, working with new technologies has become a key topic for education.

SAP has been involved in Europe-wide initiatives for the education and training of children and youth for years, including the areas of robotics (*First Lego League*) and programming technology (*Meet and Code*). Our aim is to introduce young people to new technologies in a fun way and facilitate a better start for them in their future careers.

Many teachers nowadays also want to equip their students with basic technical and digital knowledge to take with them on their journey. This does not necessarily mean that these teachers need to be experts; instead, they require practical, tried and tested working materials for learning technical skills in different subjects and at different learning levels.

In this way, we also facilitate the acquisition of digital skills in everyday school life and provide teaching materials for STEM subjects, for example. We have already initiated numerous projects to support schools with Science on Stage Germany, including the current teaching materials *Coding in STEM Education.*

I am delighted about the cooperation with Science on Stage Germany on another such goal-oriented project, and am convinced that this edition will be a complete success as well. I would also like to thank the teachers who have lent their support to this topic.

Michael Kleinemeier
Member of the Executive Board, SAP SE

<Foreword>

This booklet is special in many ways. Please allow me to me explain what I mean.

Coding is a basic skill in today's modern world and is especially important in science, technology, engineering and mathematics (STEM). Programming machines is becoming an increasingly sought-after, necessary skill in all areas of our lives and can no longer be left to IT specialists alone. Therefore, coding needs to be taught not only in computer science classes but also in every other subject. However, European curricula for STEM subjects do not regularly address this need. In the *Coding in STEM Education* project Science on Stage, the European network for science teachers, has developed teaching concepts in an attempt to close the skills gap. The overall aim of the network is to provide a platform for European STEM teachers to exchange best practice ideas. Science on Stage reaches 100,000 educators in over 30 member countries.

Some of the best teachers in Europe have contributed their ideas to this booklet. 23 teachers from seven countries met personally to exchange their ideas about coding in science education during this 18-month project. All the contributing teachers invested a great deal of time and effort in this process.

A science teacher and a computer science teacher from each of the seven countries teamed up to exchange teaching concepts with a team from at least one other country. They discussed and evaluated these concepts in their own classes in both countries to ensure that the material which you are holding in your hands is useful in real lessons, and that it has been well tested by our experts, the teachers themselves.

The focus is on programming small electronic devices like Arduino, Calliope mini, or Raspberry Pi computers. They are inexpensive devices that can solve in-depth tasks, which makes them ideal for schools.

The participants developed 11 teaching units in the fields of 'Fundamental Science in 1's and 0's', 'Microcontrolling the World' and 'Environment 4.0'. They are excellent examples of what you could do in regard to the various curricula in biology, chemistry and physics.

The *Coding in STEM Education* project was only possible due to the efforts of our enthusiastic participants, who did all this work in their spare time and in addition to their regular teaching jobs. Thank you all for this inspiring publication. Many thanks also to the other coordinators, Jean-Luc Richter, Bernd Schriek, and Sebastian Funk, who all did a great job condensing the heterogeneous thoughts and ideas of teachers from different cultural and scientific backgrounds into one homogeneous piece of work. I would also like to thank SAP SE and Gabriele Hartmann and Jens Mönig in particular. Such a publication would not have been possible without the ongoing support of these professional partners.

Coding in STEM Education is a special booklet that has been developed and tested by teachers for their colleagues in Europe and beyond. Let it inspire you to start your own projects in your classroom soon!

Dr Jörg Gutschank
Chair Science on Stage Deutschland e.V.

<Authors>

<Surname>	<Name>	<Country>	<Section>
Abad Nebot	Immaculada	Spain	Microcontrolling the World
Botelho	Lúcio	Portugal	Environment 4.0
Compte Jové	Pere	Spain	Microcontrolling the World
Fernandes	Liliana	Portugal	Environment 4.0
Funk	Sebastian	Germany	Coordinator
Georgoulakis	Georgios	Greece	Fundamental Science in 1's and 0's
Giurgea	Mihaela Irina	Romania	Fundamental Science in 1's and 0's
Gutschank	Jörg	Germany	Coordinator Fundamental Science in 1's and 0's
Hančl	Mirek	Germany	Microcontrolling the World
Ivarra	Luc	Belgium	Fundamental Science in 1's and 0's
Karagiorgou	Eleftheria	Greece	Microcontrolling the World
Lőkös	Annamária	Romania	Environment 4.0
Meier	Andreas	Germany	Environment 4.0
Mestvirishvili	Ilia	Georgia	Fundamental Science in 1's and 0's
Nicolini	Marco	Belgium / Italy	Fundamental Science in 1's and 0's
Padin	Beatriz	Spain	Environment 4.0
Poncela	Elena	Spain	Environment 4.0
Rațiu	Camelia Ioana	Romania	Fundamental Science in 1's and 0's
Reis	Jorge	Portugal	Environment 4.0
Richter	Jean-Luc	France	Coordinator Environment 4.0
Schriek	Bernard	Germany	Coordinator Microcontrolling the World
Shapakidze	David	Georgia	Fundamental Science in 1's and 0's
Toma	Corina Lavinia	Romania	Fundamental Science in 1's and 0's
Tsiliki	Sevasti	Greece	Microcontrolling the World
Tsoutsoudakis	Astrinos	Greece	Fundamental Science in 1's and 0's
van der Byl	Sonja	Germany	Environment 4.0
Winckler	Julia	Germany	Microcontrolling the World

Special thanks to Gabriele Hartmann and Jens Mönig from SAP SE for their support!
We would like to thank Holger Bach and Paul Nugent for their valuable advice during the editing process!

Coding H_2O

<Author> Beatriz Padin
<Author> Elena Poncela

<Info>

<Keywords> water, sensors, efficiency, data acquisition, environment, evaporation, condensation, solutions, mixtures, design, heat and temperature, heat conductors and insulators, solar energy, infrared (IR) radiation, reflection

<Disciplines> physics, environmental science, chemistry, computer science, mathematics

<Age level of the students> 13–15

<Hardware> Arduino[1], Calliope mini[2], Raspberry Pi[3]

<Language> Arduino[4], Python[5], block programming

<Programming level> medium

<Summary>

The students will design, build and test a solar still to purify water. They will program sensors to measure the efficiency of their solar stills.

<Conceptual introduction>

We will cover the following physics concepts:

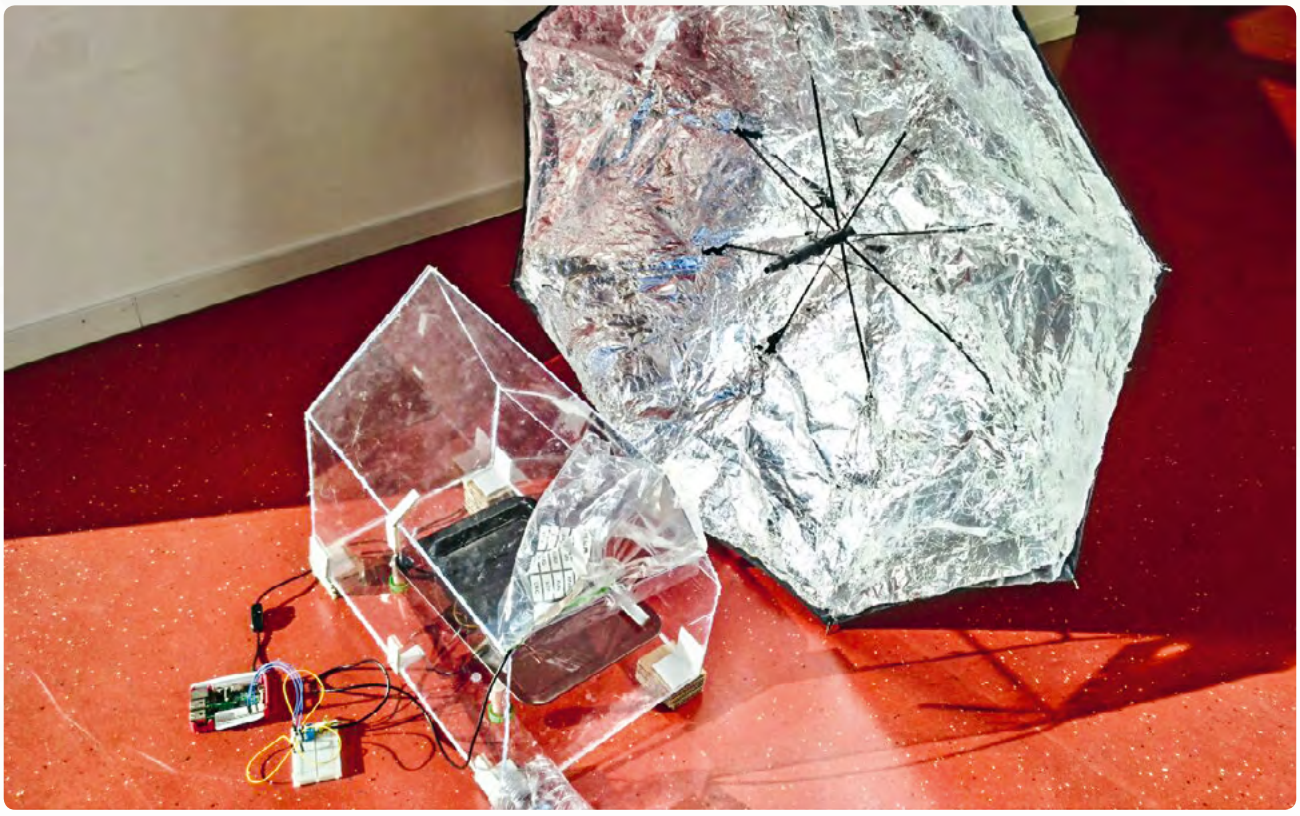
- ↳ Changes of state (specifically, evaporation and condensation) and their main characteristics
- ↳ Factors that affect the evaporation process (temperature, surface area, etc.)
- ↳ Effects of heat: changes of state
- ↳ Difference between renewable (solar energy) and non-renewable energy sources
- ↳ IR radiation and its role in transporting heat from the sun
- ↳ Radiation and its reflection on certain surfaces
- ↳ Methods for separating mixtures
- ↳ Solutions: what they are, concentration (g/L and mass percentage, etc.)

Depending on the students' prior knowledge, they will discover some of these concepts by themselves, while others will have to be explained by the teacher and experimentally validated by the students.

The aim of our project is to design and build the most efficient solar still to purify water. Only solar energy may be used. First, the efficiency of the solar still will be determined by calculating the volume percentage of the purified water that is obtained. After this, the students will program different sensors to analyse the effectiveness of their designs.

The main task of this activity is to program sensors, so the teachers will need programming skills and some basic hardware knowledge to do so.

They will need to know how to build the circuit required to connect the sensors to the microcontroller board. Depending



Solar still

on their skill level, they can either use block programming (Calliope mini[2], Snap4Arduino[6], etc.) or text programming (Arduino[4], Python[5], etc.) to program the sensors.

<What the students/teachers do>

This unit consists of three parts: designing and building a solar still, coding sensors and testing the solar still.

<Part one: Designing and building the solar still>

The project will be presented to the students and they will test their initial example of a solar still as a class by measuring the amount of cleaned water collected and calculating the efficiency of this design. While doing so, they will review physics concepts such as changes of state, solutions and solar energy.

Next, the students will be encouraged to improve this initial design. To do so, they will have to work in groups (2–3 students per group). This phase of the unit can be divided into different tasks:

1. The students will search for information about solar stills, how they work, different designs already in use, etc. During the research process, they will be asked to reflect on the following questions:
   a. Evaporation process: what are the main factors affecting this process? Think about the surface where you are going to place the dirty water. Is it better to have a wide or a narrow surface, a larger depth or a smaller one? Does the colour of the container matter?
   b. Condensation process: what is necessary to produce condensation? Do you need to design a large or a small surface for water condensation? How are you going to make the clean water move to the point where you wish to collect it?
   c. Radiation process: how can you maximise the IR radiation hitting your solar still? How can you reach the maximum temperature possible in your solar still? Consider using a surface covered with aluminium foil to reflect the sunlight into the solar still.
2. The groups will create their designs and explain them to the teacher.
3. Once the teacher has approved the design, the students will be asked to find the appropriate materials (at school, at home, order online, etc.) and build their still.
4. The students will determine the efficiency of their solar stills without sensors. Using a graduated cylinder, they will not only measure the volume of dirty water but also the volume of clean water collected and apply the following mathematical equation:

$$Efficiency = \frac{volume\ of\ water\ collected}{volume\ of\ dirty\ water}$$

A step-by-step guide including tasks and questions for the students is provided for the second and third parts of this activity.

The aim is to give the students different choices with regard to sensors, programming languages and hardware, but this also depends on each individual school/class (available materials, programming language knowledge, etc.).

<Part two: Coding sensors>

Only the best solar stills will be tested using sensors. In this part, you will need to:

1. Select the microcontroller board, programming language and sensors that you are going to work with. Answer the following questions to help you make the best decision:
   a. Will you use block programming or text programming? If you choose block programming, consider using a Calliope mini[2] or, alternatively, program a Raspberry Pi[3] with Scratch[8]. In case you opt for text programming, you could use an Arduino[1], or code in Python[5] for Raspberry Pi[9].
   b. Are you going to use digital or analogue sensors? If you decide on the latter, Arduino could be the best choice.

2. Choose the sensors that you want to use. To keep things simple, please select no more than two sensors. Some examples are temperature, humidity, rain and IR radiation. Before making your decision, consider the specifications of each sensor. Is the output signal analogue or is it a digital signal with only two possible values (true/false)? How does the output signal relate to the value of the parameter that you are measuring? Are they directly proportional? Does the output increase as the parameter decreases?

3. Build the circuit to connect your sensor to the microcontroller board. Use the additional resources provided[7] or search the Internet for examples.

4. Code the sensors. You must follow these steps:
   a. Write down what you want your program to do, considering the features of the sensors that you have selected. Do you want your program to only display the value of the measured parameter? Do you also want to show the maximum and minimum values? Does your sensor show the actual value of the parameter or do you need to make any calculations?
   b. Write your program. Do not forget to write comments on your code. You can use the additional resources provided by your teacher as a guide.[7]

Flame sensor

5. Test your code. Does the program work as expected? Examples:
   - ↳ You can measure the amount of infrared radiation that reaches the solar still with a flame sensor and an Arduino UNO[1]. If you are using a surface covered with aluminium foil, use this sensor to test whether the radiation is reflected into the solar still.
   - ↳ You can record the maximum temperature and the relative humidity reached inside the solar still with the DHT11 or DHT22 humidity and temperature sensor and your Arduino UNO.
   - ↳ You can write a Python 3 program to determine the time that it takes for the first water drops to condense with the FR-04 rain sensor and Raspberry Pi[3]. You can also use this sensor with a Calliope mini and program it with Snap![10] (block programming language).

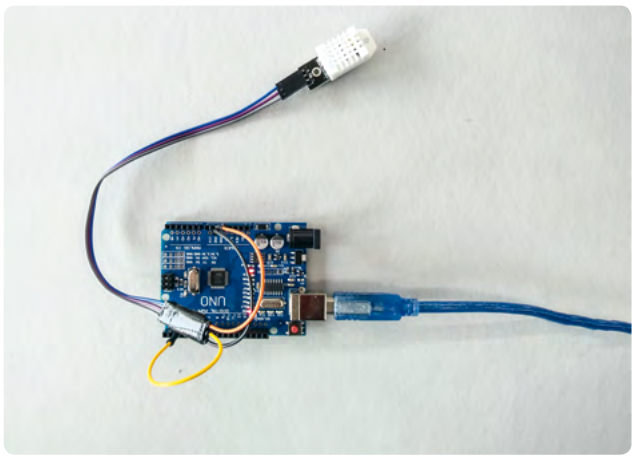
<Part three: Test the solar stills with sensors>

Your team has to use the programmed sensors to test and compare your design with another one. The solar stills have to work under the same conditions and use the same type of sensors for this to happen.

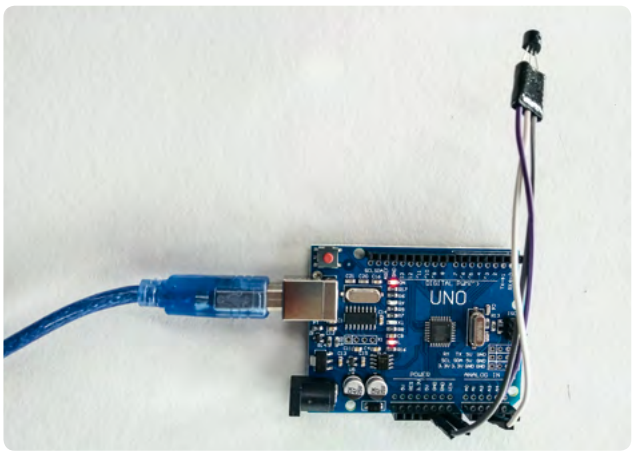
You will analyse the reasons why the efficiencies are different, noting key facts like the factors that affect evaporation, etc.

How to improve the efficiency of the solar still:

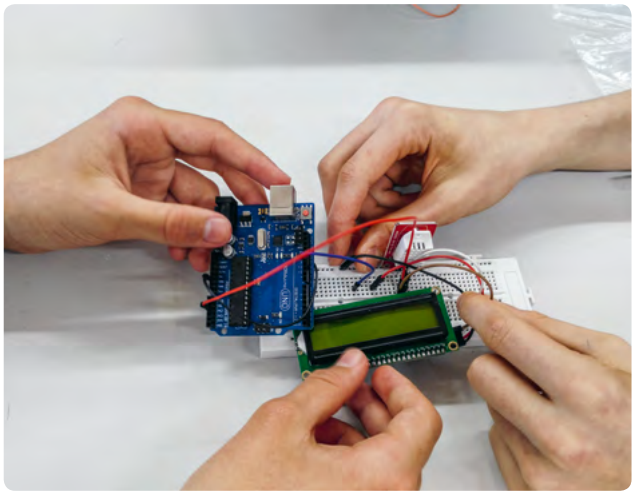
- ↳ Test it on a sunny day over the midday period.
- ↳ Allow your equipment sufficient time to reach the highest temperature possible.
- ↳ Make sure the solar still is airtight to avoid any loss of water vapour.



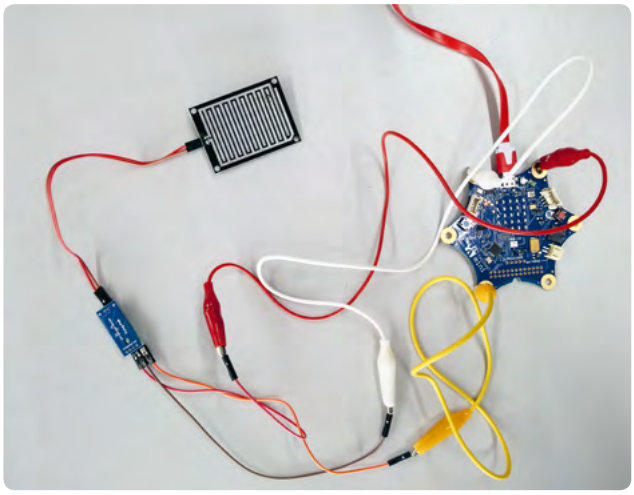
Arduino with humidity sensor



Arduino with temperature sensor



Arduino with LCD display and humidity sensor



Rain sensor with a Calliope mini

- Remove the clean water on an ongoing basis to prevent it from evaporating.
- Colour the dirty water to ensure that the solar still works properly.
- Choose a wide, black container for the dirty water.
- Use an umbrella covered with aluminium foil to reflect sunlight into the solar still.

<Results>

The first part of the activity (building the solar still) led to designs that varied greatly in their efficiency. When tested on sunny days during May/June in Spain, the best design purified 95 % of the dirty water in 24 hours. A 54 % efficiency rate was also obtained in 4 hours. Nevertheless, some stills collected no clean water at all due to various deficiencies in their design.

When applying the sensors, the students obtained the following results:

- The maximum temperature reached inside the solar stills after one hour on a sunny day was 65 °C.
- The influence of the colour of the container on the dirty water was analysed. Upon comparing a white and a black container left in the sun for a few minutes, the temperature of the water in the black container was found to be almost 5 °C higher than in the white one.
- The relationship between the temperature of the water and the rate of evaporation was studied using the humidity sensor. The relative humidity obtained inside a solar still that contained water at room temperature was measured at 55 %. When the water was heated to 45 °C, the relative humidity increased to 98 % in just a few seconds.
- The amount of radiation reflected by a metallic surface into the solar still was measured with the flame sensor. An old umbrella covered with aluminium foil was very effective in reflecting IR radiation.

<Conclusion>

The students will use their creativity and higher-order thinking skills (HOTS) to design the most efficient solar still. This unit will also give them the opportunity to develop their critical thinking and problem-solving skills, which is very useful because the students can utilise them on a daily basis. They will learn some key physics facts (self-learning) by observing, experimenting, testing and analysing their results, instead of simply reading about them in their physics textbooks.

They will also develop computational thinking skills while coding their sensors. They will be doing physical computing (i.e. their codes will interact with the physical world). Throughout the unit, they will follow the different steps of the scientific method outlined above to develop the best solar still design and then build it with suitable materials.

It is important that every student is involved in the project tasks. Some tasks can be done individually (searching for information, collating their initial ideas about the design…) to ensure that this happens.

However, the main obstacle is that some students may not have the appropriate programming skills (which is why we have included block programming alternatives), sufficient knowledge to build the circuits (the additional resources[7] are very helpful in this regard) or may not understand how the sensors work. In addition, the materials may not be available in all schools, so they may need to be bought.

Extension activities:

- The data collected from the sensors could be stored on an SD card for further analysis.
- An LCD screen could be used to visualise the measurements.

↳ Internet of Things (IoT): the data collected could be sent through the Internet in real time so it is publicly available.
↳ Additional sensors could be used, such as CO_2 or other greenhouse gas sensors, a conductivity sensor to check whether the clean water is still salty, a pH sensor to measure the pH of the dirty and clean water, etc.
↳ The salinity could be measured with seawater samples.
↳ A method to disinfect the collected water could be added.

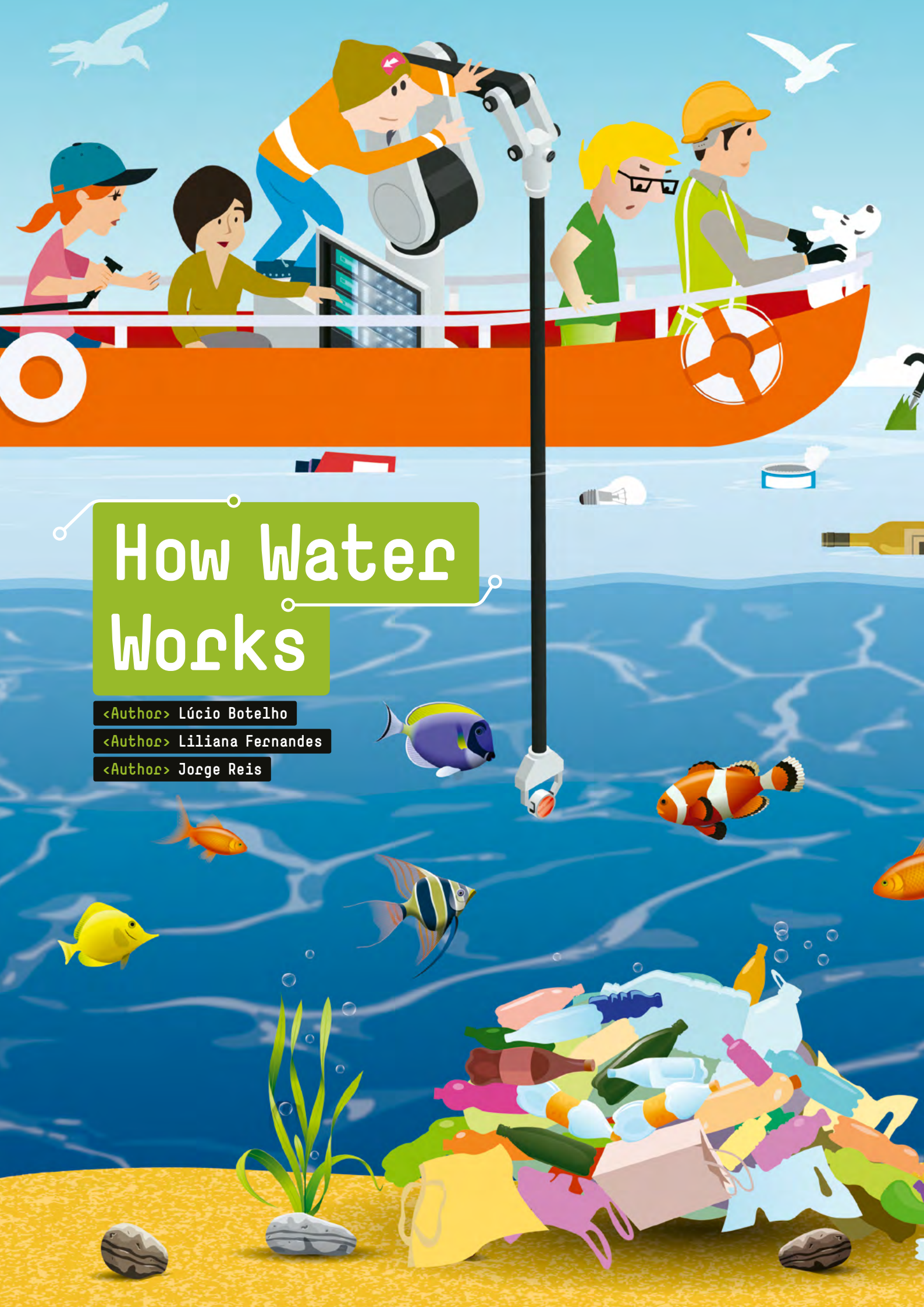
The students could use the solar stills to study the greenhouse effect, photosynthesis, cell respiration, ideal gases, etc. Many sensors could be used in other projects to measure physical or chemical parameters; for example, to monitor air pollution or water quality.

<Cooperation activity>

Are solar stills more efficient in countries with sunny weather? Students from different European schools could share their results, using an online map for location purposes. The salinity of the different places could be compared by taking seawater samples.

<References>

[1] www.arduino.cc/
[2] https://calliope.cc/en
[3] www.raspberrypi.org
[4] www.arduino.cc/reference/en/
[5] www.python.org/
[6] http://snap4arduino.rocks/
[7] All additional materials are available at www.science-on-stage.de/coding-materials.
[8] https://scratch.mit.edu/
[9] www.raspberrypi.org/documentation/usage/python/
[10] https://snap.berkeley.edu/



How Water Works

<Author> Lúcio Botelho
<Author> Liliana Fernandes
<Author> Jorge Reis

<Info>

<Keywords> water, image processing, data acquisition, microclimate, robots

<Disciplines> mathematics, biology, social studies, robotics, arts

<Age level of the students> 6–10, 11–15 and 16–18

<Hardware> <easy level> Calliope mini[1], LEGO We Do 2.0[2], small learning bots[3], WeeeMake[4]
<medium level> LEGO EV3[2] with LEGO ultrasonic and colour sensors, or Anprino[5] with Arduino[6] and appropriate ultrasonic[7] and colour sensors[8]
<advanced level> computer with Internet access

<Language> Snap![9], Scratch[10], WeeeCode[4], Open Roberta[11], LEGO Blocks[2]

<Programming level> easy, medium, advanced

<Summary>

This unit was designed to be transdisciplinary in nature, i.e. facilitate collaborative work between students of different levels, from primary to secondary school age. Alternatively, each part can be taught at its own level individually. Starting with a computational thinking approach, to coding in Scratch[10], through to programming robots and an ecological house, in the teaching unit 'How Water Works' students will discover everything about the topic of water.

<Conceptual introduction>

This project is all about water, its role in our life and our role in preserving it. Divided into three levels (easy for primary school students, medium for middle school students, and advanced for secondary school students), this project can be adjusted for collaborative work at different school levels and in cross-curricular activities.

<What the students/teachers do>

<Easy level: Where does water come from?>

The students will be challenged to investigate where water comes from. The teacher will ask questions to stimulate the students' interest and then … the adventure will start! The students will research, learn and then share their findings with their classmates. At the same time, the students will start to develop their computational thinking skills with easy challenges that teach them how to program simple bots.

After finishing their research, the students will start to work in small groups and build some beginner projects using the demo modes of the WeDo 2.0 app[12] with special emphasis on water-related tasks.

Then the students must build an ecological solution, using any bricks or set they want, in which they present an innovative approach to save water.

In the example below, the students built an ecological house[13] and combined it with some extra bricks and the WeDo 2.0 set. Then they added a rainwater collector connected to a filter (coded with the LEGO app) that directed the water into the farm so the animals could drink fresh water (📷1).

📷 1: An ecological house

At the same time, the students, still working in small groups, will start to plan and design new mats related to water for small learning bots that could be programmed without a computer. By presenting their mats to other students, they will motivate them to code and learn about water at the same time. The students can use various low-cost learning robots to do the task.[3]

Full instructions on how to print the mats are available online.[14]

📷 2: Students designing mats

You will also find a link to a step-by-step unit plan in the additional online material.[14]

<Medium level: Building a water dam cleaning robot >

The dam cleaning project is about a robot that travels through the water reservoir created by a dam and detects solid waste.

This project has two versions, using two different robots. The LEGO version (📷3) uses the EV3 LEGO Education kit; the Anprino robot[5], which is printed using a 3D printer and then assembled, is the work base in the Arduino version (📷4). The Arduino microcontroller[6] and its range of accessories are attached to the Anprino.

📷 3: The LEGO version

📷 4: The Anprino version

Start by building the water reservoir model using paper or cardboard. It should measure about 2 m × 1 m and be painted blue to simulate water. Build the shores using strong cardboard to limit the robots' room to move, and simulate the waste with pieces of black cardboard.

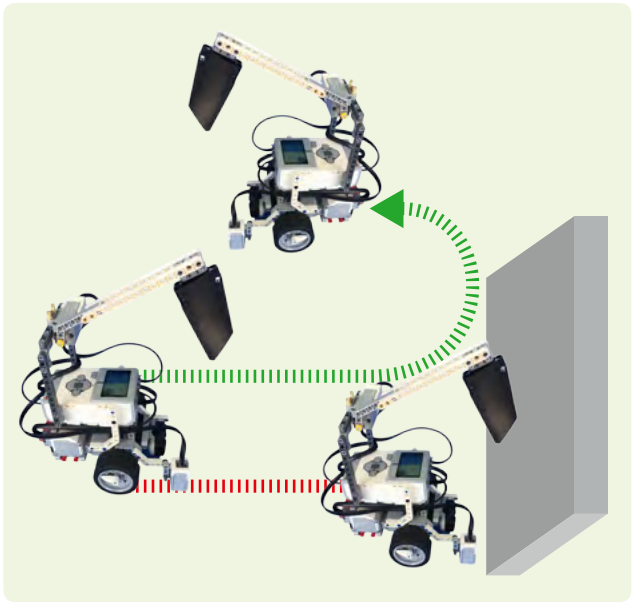
📷 5: Model of the dam reservoir

Ultrasonic sensor

An ultrasonic sensor [7] generates sound waves to detect and measure the distance to the objects. It can also send sound waves to function as a sonar or receive a sound wave that starts a program mode.

Using the ultrasonic sensor, the robot can detect obstacles and react in different ways, depending on the code. The robot could be programmed to stop or change direction, for example. In the dam model, the obstacles are the cardboard shorelines.

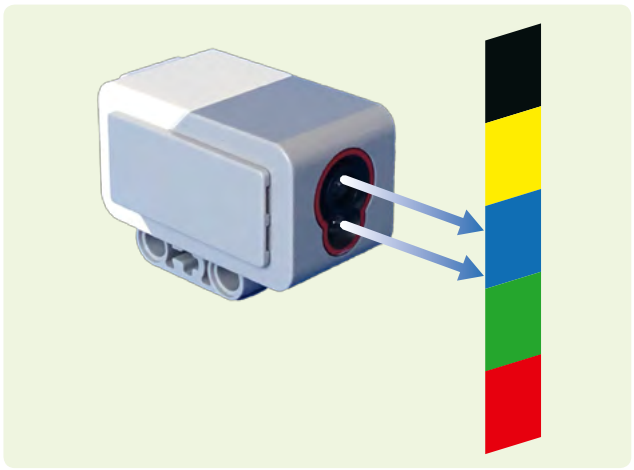
📷 6: A LEGO ultrasonic sensor



📷 7: Robot stops/Robot changes direction

Colour sensor

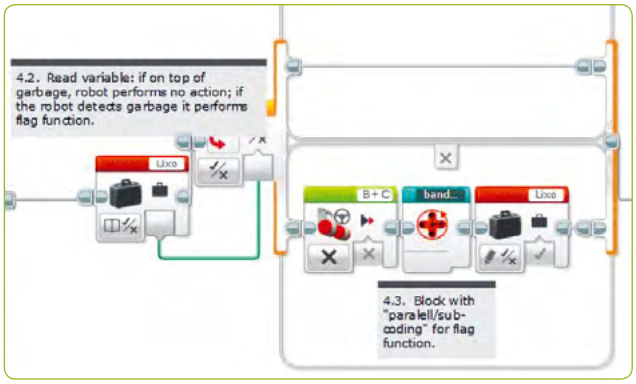
The colour sensor[8] can detect different colours and the absence of light. It works as a light sensor as well by detecting varying intensities of light. Students can build different colour lines for the robot to follow.



📷 8: An example of a colour reading sensor: It distinguishes colours by reading their RGB code.

Building code, using LEGO programming blocks

The students must build different models to simulate distinct kinds of waste, such as domestic, industrial, tourism-related, organic, etc. The main goal is for the students to become aware of river and dam pollution. The students must simulate a waste detector assembled in a boat and then plan and build a waste collector boat at a later date.



9: An excerpt from the LEGO programming; the complete diagram is available online[14]

The robot must play a particular sound for every type of waste that it detects. To achieve this, we use the colour sensor and specific colour 'stains' that simulate each type of waste.

The students can consult the publications of environment/government organisations and build the different stains according to pollution statistics.

The students will go on field trips to rivers and dams to examine the water quality and any pollution. They must then simulate these observations in the models that they build. With the help of the robot, they should scan and note the results in a table (10).

When the students have collected enough results, they must present their research to the class. The goal is for the students to develop their critical thinking, investigation and coding skills. When the students look at what is happening in our waterways, they will see the consequences of centuries of blindness to environmental issues. To be able to understand this, they must have acquired the requisite environmental skills to intervene in their community. For example, this could be by alerting people to the need to prevent behaviour that damages the environment and water in particular. Additionally, they must also be able to plan and present solutions when they detect problems. The overall goal is to increase their civic participation and sense of environmental responsibility in their community.

Please note: our students have already built and tested the LEGO version and are still in the process of improving the Arduino version. The complete code used to program the Arduino is available online.[14]

<Advanced level: Programming educational environment-related games>

The main goal is for the students to become aware of water pollution. The students will use Scratch[10] to program games that motivate others to help to preserve and protect water and thus encourage people not to throw waste into bodies of water.

Our first game simulates a little fish in the ocean. The fish has to feed while, at the same time, avoiding other marine creatures (sharks and crabs) and falling waste (glasses, cans, etc.). The more it eats, the bigger it gets, and the more points the player earns in the process.

The fish must not collide with waste and other fish or it gets hurt and receives a bandage. When it has three bandages, the game is over. This game is funny and not only sensitises children but also adults to the increasing amount of waste in our waterways (oceans, rivers, etc.).

The second game is based on a well-known video game, where a frog has to cross a street. But in our case, the character has to cross a river (using the logs as the water flows quickly) and avoid rubbish as well as other animals (bats and snakes). It can also eat flies to gain extra points.

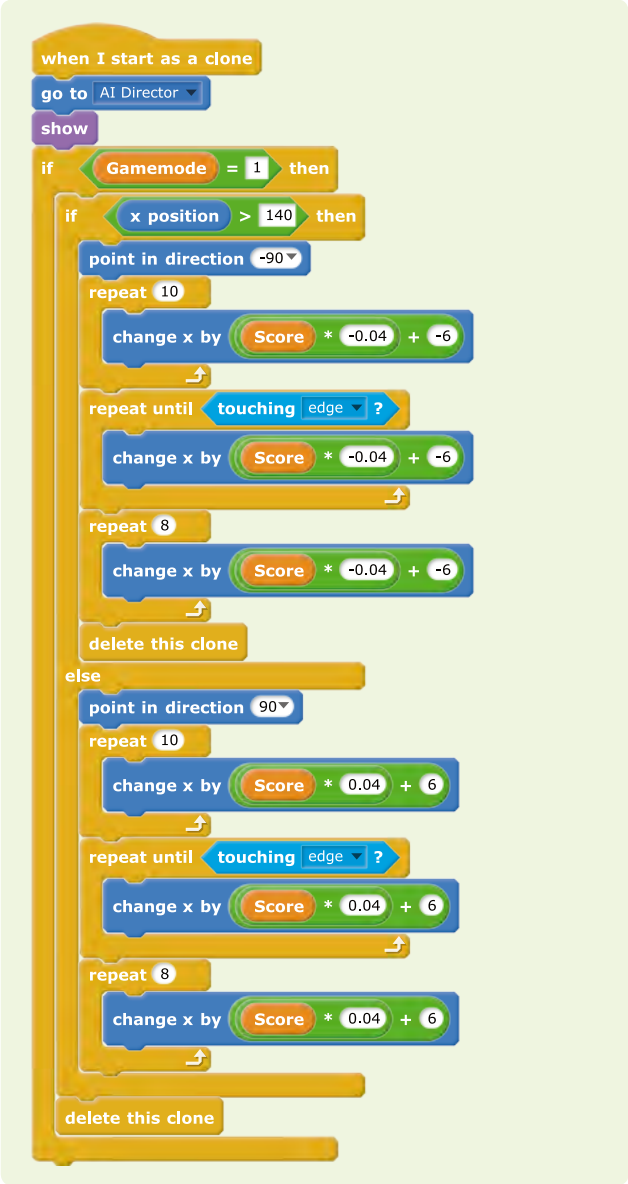
10: Table for waste scan, data related to two different excursions and waste collected in each one by the cleaning teams from the school's environment club

Date	Type of waste					Cleaned Area
	Domestic	Industrial	Undifferentiated	Organic	Others	
Excursion April 2018	3.450 kg			32 kg	8 kg	100 m²
Excursion May 2018	0.730 kg			6 kg		100 m²

In this game, there are four different scenarios and one is selected randomly at the beginning of each round. The frog (sprite) has three lives, after which the game ends.

The following section contains details about the program.

📷11 shows the part of the program that controls the movement of some of the enemies in the various game modes. In the displayed example, the enemy disappears when it touches any of the edges. As long as it is not touching the edges, it repeats the same movement, which also increases in speed with a 0.04 factor adjustment as the player's score increases. This is a very clever way of making the game a little more challenging as the score increases with the increasing level of difficulty.

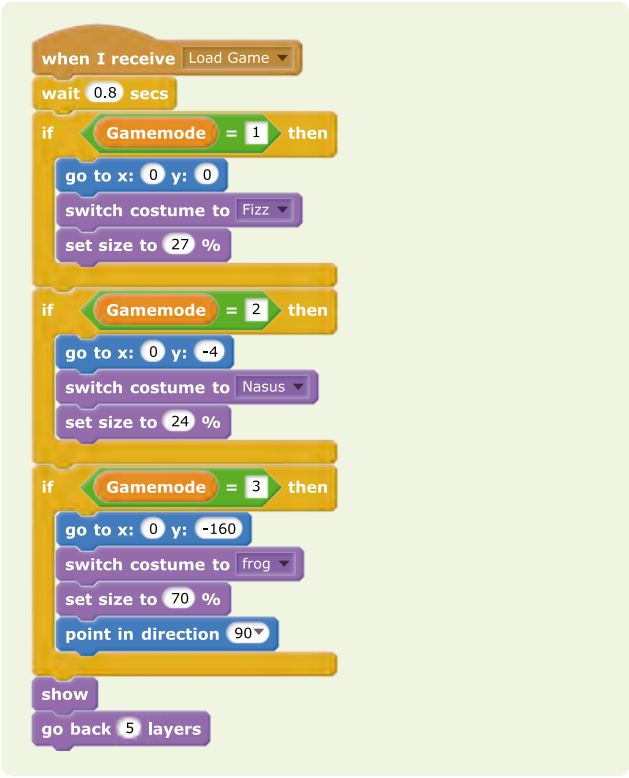


📷 11: Scratch program enemy control

Game start: choose one of three game modes (📷12). At the moment, two games are ready and the students are developing a third one called Game mode 2.

For example, Game mode 2 could be in a pond where ducks have to catch some food.

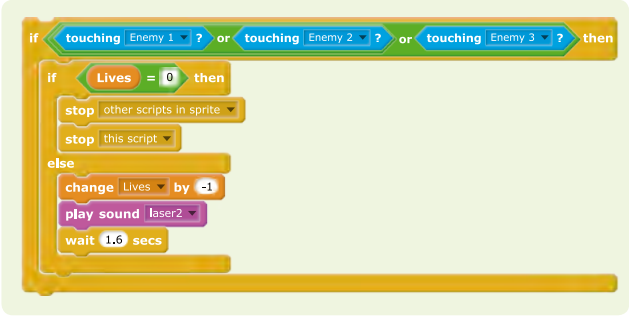
Ducks regularly eat small fish and fish eggs, snails, worms, molluscs, and small crustaceans such as crayfish, grass, leaves, weeds, algae, aquatic plants, roots, small frogs, salamanders and other amphibians. Additionally, the ducks must try to avoid other ducks or waste in the pond (or in advanced levels, random poachers).



📷 12: Scratch program game start

If the fish touches any of the enemies (1, 2, or 3), it loses one life and a sound is heard.

If the player loses all of his/her lives, the game is over, i.e. all the scripts are stopped (📷13).



📷 13: Scratch program enemy 1-3

The game is very well programmed and constructed because the same code is used for the two versions of the game. The same sprite changes its costume from 'Shark' to 'Bat'.

The students will all learn to improve this program by providing new ideas or helping with innovative coding solutions so that the code can be even better and more fluid.

This is possible because the games have similar goals:
↳ avoid enemies
↳ catch food/flies
↳ losing 3 lives means the game is over
↳ earn points (the fish by eating fish food, the frog by eating flies and reaching a new scenario)

They will use clones of the sprite enemy so that they can make the same sprite appear from different directions and have different behaviours (directions) in the game.

The full program is available for download.[14]

<Conclusion>

In this unit, the students will work collaboratively with their peers and their community, and learn and share their knowledge about water: water cycle, water shortage, pollution, etc. They will also develop resources to monitor, save and protect water. At the same time, the students will develop investigative tools and coding skills, as well as skills in the field of robotics. When the oldest students mentor and support the younger ones, they all will motivate and challenge each other to advance their work. This contributes hugely to the success of the projects.

At the end of this school year, we noticed that the students had not only improved their programming skills but were also more conscious of water problems and the dangers for animals and plants, which depend on clean water for the safety of their habitats.

It is not easy to code several games in one. The games must have some similarities so that the code from one can be adapted to serve all modes. However, it is a clever way to save on coding resources.

We chose to work together, although in different schools that are far away from each other, because it allowed us to share ideas and improve the collaborative work between students of varying (social and economic) backgrounds and ages. It was not easy to meet face-to-face or to get the students together as much as we intended, but it turned out to be a good option as it allowed the students to share their ideas and methodologies and to interact with unknown peers, which improved their communication skills. It also offered them the possibility to participate in different competitions and to discuss their results and come up with improvement tips with other students. An alternative to personal meetings could be to communicate online via video conferences. Finally, the students were able to share their work with the community and play a part in changing local attitudes towards water protection.

From this unit, you can challenge your students to develop other ideas and concepts on the topic of saving water, and contribute to improving environmental behaviour in your community and thus reducing the students' and hopefully the community's carbon footprint in the process.

<Cooperation activity>

'Science on Stage is about sharing resources among teachers!'

As a result of this project, a community of teachers was strengthened and resources as well as ideas were shared. This has contributed to better student learning throughout Europe. Sharing and collaborating is the best way for us all to improve and further develop these projects.

<References>

[1] https://calliope.cc/en
[2] https://education.lego.com
[3] Possible bots: Bee Bots from tts, DOC from Clementoni, Jack from Imaginarium
[4] www.weeemake.com/
[5] Anprino is a robot developed by the Portuguese National Association of Teachers of Information Technology (ANPRI); information and 3D printing files www.anpri.pt/anprino/index.php/anprino-luis (29/11/2018)
[6] www.arduino.cc/
[7] We used the HC-SR04 ultrasonic sensor.
[8] We used the BE15000624 light sensor.
[9] https://snap.berkeley.edu/
[10] https://scratch.mit.edu/
[11] https://lab.open-roberta.org/
[12] https://education.lego.com/en-us/downloads/wedo-2/software (29/11/2018)
[13] LEGO SET 31068
[14] All additional materials are available at www.science-on-stage.de/coding-materials.

VPLS – Vacation Plant Life Saver

<Author> Andreas Meier
<Author> Sonja van der Byl

<Info>

<Keywords> automated plant watering, using a microcontroller to control a water pump

<Disciplines> computer science, natural sciences, technology

<Age level of the students> 10–14

<Hardware> computer (one per student if possible), Calliope mini[1] with a humidity and temperature sensor (one per group)

<Language> Scratch[2] (online or offline), editor for Calliope[3] (online)

<Programming level> easy

<Summary>

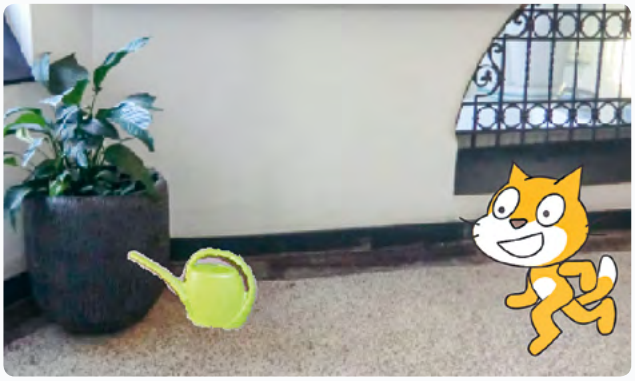
The potted plants in school buildings often die during the summer break because nobody takes care of them—this is why we need a Vacation Plant Life Saver. In this teaching unit, we will develop a virtual and a real-life lifesaver for school plants.

<Conceptual introduction>

The project is suitable for all STEM subjects because the level of programming required is basic. In the first section, the watering of plants at regular intervals is realised in virtual terms. This will cover various control structures in computer science such as object orientation, loops and conditions as well as the use of variables. The students will do some basic coding for the first time and work through a tutorial online in 'Getting Started with Scratch'[4] (duration: 45 minutes).

Object Oriented Programming (OOP) is a type of programming which lends an object properties (attributes) and capabilities (methods). In our case, these objects are a cat called 'Sprite', a watering can and a stage. Every object belongs to a class. All the objects in the same class have the same properties and capabilities. You can interpret a class as a blueprint and an object as an instance, i.e. a concrete realisation of the blueprint. In Scratch, figures are instances of the 'Sprite' class, or in short, 'sprites'. The cat called 'Sprite' is a sprite, i.e. an instance of the 'Sprite' class.

One of the attributes of a sprite is its costume; in this case, it is the image of a cat. Another instance of the 'Sprite' class could have the image of a human as a costume and be called Gunther. The cat 'Sprite' and the human 'Gunther' are both objects (instances) of the 'Sprite' class or in short: both are sprites. There are only two classes in Scratch: the stage and the sprites. In this project, we have two sprites (the cat and the watering can) and the stage (📷1).



📷 1: The virtual Vacation Plant Life Saver

The second part of the project can be done after the first part has been completed, or it can also be carried out independently if the students know the previously mentioned control structures and have already gained initial programming experience with the Calliope mini[1]. Microcontroller sensors which control the valve of a water pump will be used instead of variables.

<What the students/teachers do>

All the required materials and worksheets are available for download.[5]

<Part 1: Virtual regular plant watering>

Step 1: Coding a program with only one variable, time; getting to know one-sided conditions and loops (duration: 180 minutes).

After analysing the problem ('How could a virtual Vacation Plant Life Save work?'), the students will be asked to think about the basic structure of such a program. They will then make notes in form of a recipe (algorithm) and test each other's ideas. Only after doing so will they agree on a common basic structure for the program (see Worksheet 1[5]).

This phase will help the students to think about the basic structures of the program that they want to code. The keywords 'list of statements', 'loop' and 'condition' are derived from the context.

Now the students will realise the program in Scratch[2] by assembling the individual components[5] provided into a working program. The students will learn more about Scratch and the following aspects in particular in the process:

- ↳ object orientation (each figure has its own script, even the stage)
- ↳ structure (What does the structure of a condition or a loop look like in Scratch?)
- ↳ script/costume/sounds can be assigned to each individual figure

Additionally, the students will learn more about the basic structure of the plant watering program while thinking about how best to order the individual parts of the code to make the program work. It is especially important to decide which statements need to be inside the counting loop (📷2 & 3); this can be done by trial and error.

📷 2

📷 3

Based on the program assembled in the section prior to this, the students will now be able to create their own program in which the cat 'Sprite' moves to the plants and waters them according to the variable time. The only file that the students will be given is the stage for the starting scene.[5]

The students will be encouraged to investigate the programming language independently to test out their own ideas and be creative. It is important that the students are able to use the requisite programming language with confidence (according to their respective level of knowledge); this way it will be more enjoyable for them.

Step 2: Write a program that incorporates the 'water level' and 'temperature' variables (required time approx. 270 minutes).

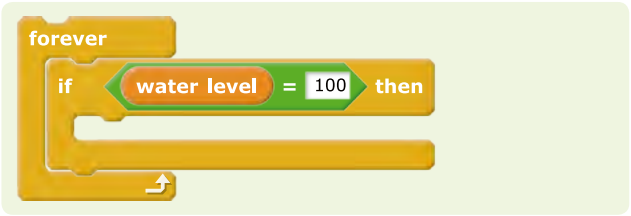
As an introduction to the second step of the teaching unit, the students will think about other factors that determine how often a potted plant must be watered. The 'room temperature' and the 'water level' in the plant container will certainly play a role here as changing variables (see Worksheet 2[5]).

The students will receive a working program in which the scripts for the cat 'Sprite' and the watering can are nearly the same as before.

However, there will be a new script for the stage which controls the 'water level' variable in place of the previous timer. The cat will only water the plant once. The students will be encouraged to think about how the 'water level' variable can be defined on the one hand and how it controls the activity of the cat. On the other hand, they will be tasked with solving the problem so that the plant is only watered once. There is a help file available if required.[5]

By using only one variable, the program will stay clearly structured. It might be too challenging for a beginner programmer to coordinate several variables at once.

The loop structure that is used is more complex than before as it is connected to a condition (📷4). The students will need to think carefully about what needs to be repeated, how often and under what conditions.



📷 4

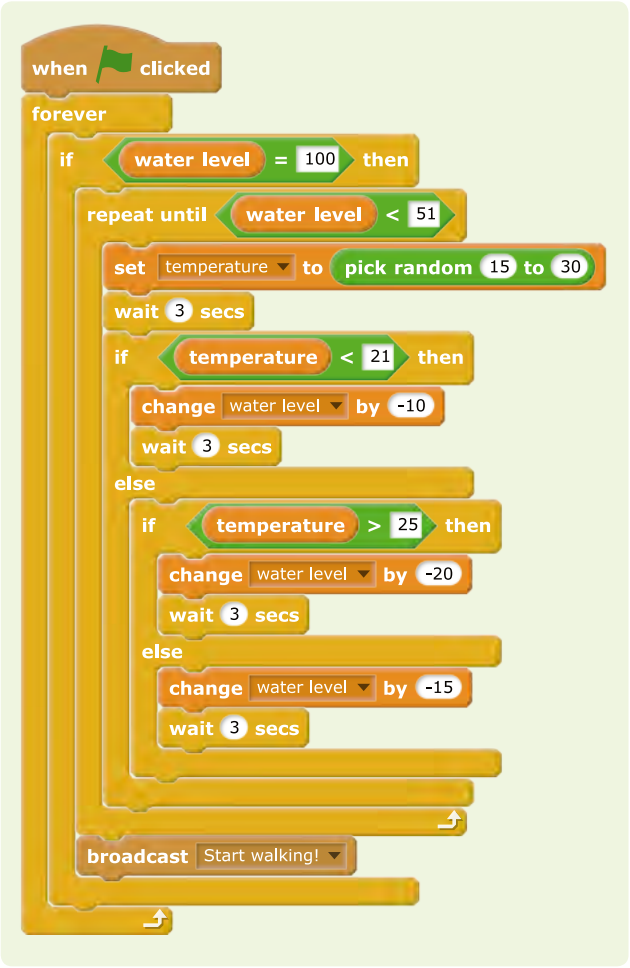
The ability to structure is fundamentally important in learning any programming language, but it will be taught in a very fun way; the students will be able to try out everything without any negative consequences.

Fast learners will also have the opportunity to change the program and try out their own ideas at the end of this work phase.

In the next step, the 'temperature' variable plays a role in the program. Its value is determined by a random number generator, which provides a number between 15 °C and 30 °C. The water level in the plant container changes, depending on this value (Worksheet 3[5]).

Since the program structure of the 'new' watering program is quite complex, a disassembled program should first be re-assembled into a functioning program[5].

The random number generator and two-sided conditions will then be introduced. In addition, the students repeatedly work with variables as well as query conditions and deepen so their understanding of them. Again, the students will need to think carefully about what needs to be repeated, how many times and under what conditions (📷5).



📷 5

As in the previous step, the students will be given the opportunity to customise the resulting program by working according to their ideas and programming ability.

At the end of the first part of the project, where the Vacation Plant Life Saver was realised in virtual terms, a wide variety of programming results should be presented to acknowledge the ideas of the students and recognise their performance.

<Part 2: Regular watering of a plant controlled by a microcontroller>

During the project, it quickly became clear that some students were not satisfied with the virtual solution to the watering problem. They asked about ways to water real plants with the help of their program. The virtual watering program is relatively easy to transfer to a real VPLS using a microcontroller, especially as many of these mini-computers can also be programmed with Scratch[2] or a similar app. In our project, we use the Calliope mini[1], a microcontroller similar to the BBC micro:bit[6], which also contains user-friendly 'plug-and-play functions' such as touch sensors and motor connections. However, the VPLS can also be controlled with all other microcontrollers commonly used in schools, such as LEGO EV3, LEGO NXT, Arduino, Raspberry Pi, Teensy, etc.[5] It is a simple process to program the Calliope mini, as you only need to connect it to the computer via a USB cable. Open Roberta Lab[7], which supports various microcontrollers (alternative editors are available[5]), is a suitable programming interface. The programming interface is available in different languages and you can change the language by clicking on the globe icon after you selected a microcontroller, in our case the Calliope mini.

A simple version of the VPLS could work as follows:

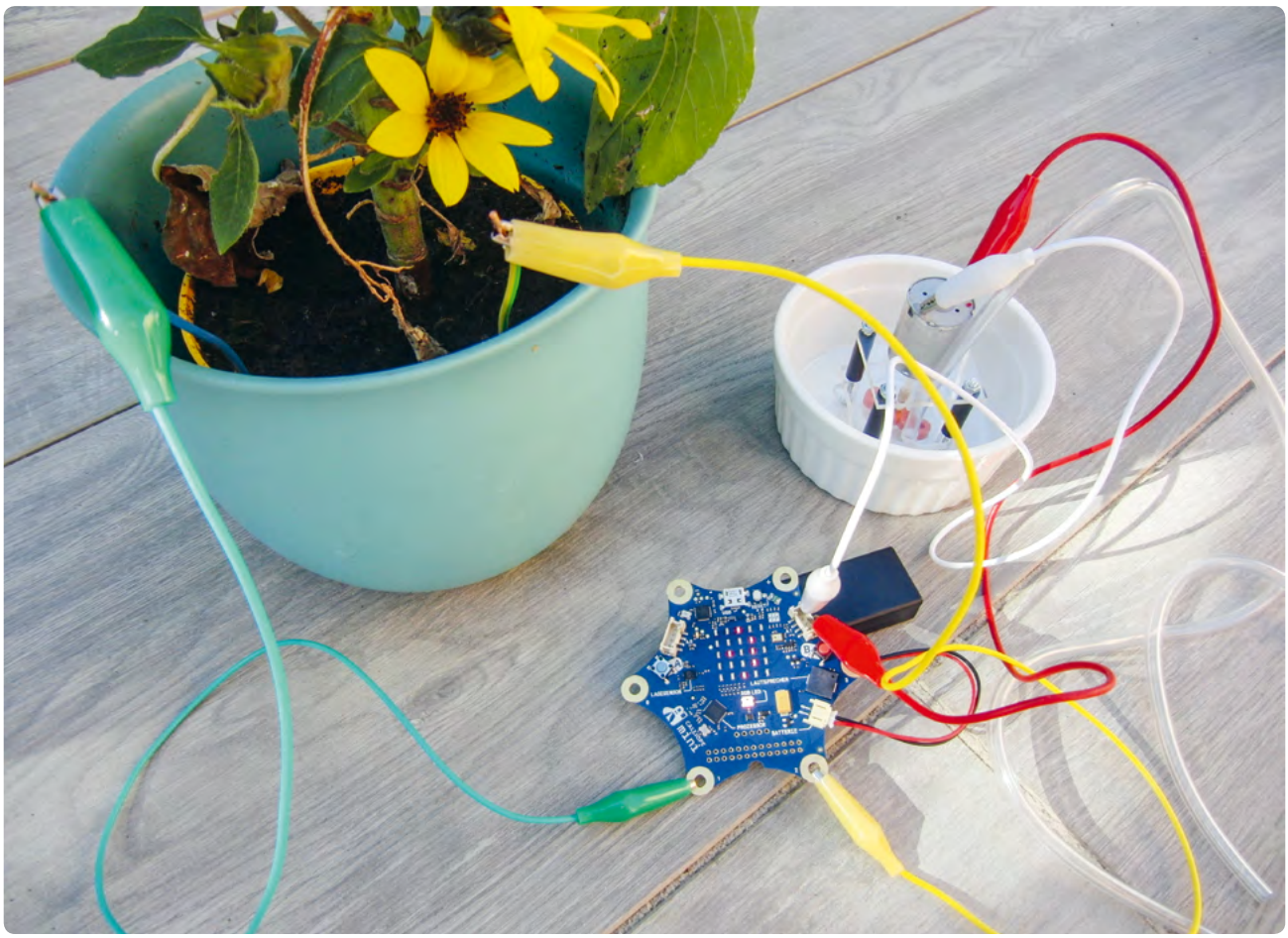
1. The humidity of the soil is constantly measured.
2. If the soil is too dry, a certain amount of water is pumped in until the soil is humid enough.

Therefore, the microcontroller must be able to measure the soil moisture and control the motor of a water pump.

The Calliope mini has four touch sensors. The physics behind them is that they measure the electrical conductivity between the connection points. As water conducts electricity, humid soil has a higher conductivity than dry soil. You simply use two copper wires as sensors, which you then put into the flowerpot some distance apart. They are then connected with crocodile clips to the Calliope mini with the contacts at the corners (–, P1). The output value of the sensor P1 (analogue pin) will be between 0 and 1023. If the conductivity decreases to a certain value, the pump that waters the plant will be activated (see 📷6).

An essential component of the pump is a small electric motor, which is connected to the Calliope mini either directly or with the help of an additional motor driver, depending on the level of power required. Two motor ports (A, B) are available in the Action/Move menu. The water quantity of the pump can be adjusted by changing the 'speed %' value.

A certain level of craftsmanship is of course required to construct the pump and to build the motor driver. We provide the requisite construction manuals in the online material[5]. The pump costs just a few euros.



6: VPLS controlled by a Calliope mini

While they are working with the real VPLS, the students will come up with numerous questions on how to optimise it, which will deepen their knowledge in the process; for example:

- ↳ Do our plants need a lot of water, or just a little? How large does the reservoir need to be?
- ↳ What is the best time to water the plants? And, is it better to water a lot only once a day or less but several times a day?
- ↳ At what depth should the humidity sensors be placed in the soil to provide optimum measurements? What is the ideal distance between the sensors?
- ↳ How long does the power supply of the VPLS last? Can the energy efficiency of the pump be increased so that the VPLS waters during the entire school break?

As you can see, the VPLS project offers the students various approaches from the fields of biology and physics for future experiments or project work. Another interesting option would be to connect the VPLS to the Internet and monitor online (Internet of Things). Although this would go far beyond our introduction to programming with the VPLS, it shows what can result from a simple question.

<Transferability to other programming languages>

The project can easily be transferred to Snap![8], which is a further development of Scratch[2]. Programming examples are provided online[5]. These two programming languages are particularly well-suited for a project that is aimed at beginner programmers, because they are easy to understand and encourage the students to try out ideas by using 'drag and drop'.

<Conclusion>

Students who are new to programming will take their first steps into the field and learn important basics of a programming language in the course of this project, which has its roots in a real-life situation. The focus is not on learning the syntax and vocabulary of a programming language, but rather on trying out the effects of certain structures: 'What works and why?'.

Errors are welcome because they are usually easy to find and to explain, and thus help the students to understand how a programming language works.

On the one hand, the given framework gives students security (whoever is unsure will only solve the puzzle of assembling

the provided program parts), and on the other hand, there is plenty of room for creativity for those students who do the 'mandatory tasks' quickly.

Some ideas have emerged from the project, e.g. to build a 'real' watering robot or to develop a watering game. In any case, the students gained positive initial programming experience that hopefully will have a lasting and sustainable effect on them.

The time frame set at our school was sometimes difficult. We only had 45 minutes to work on this project per lesson. The organisational part of the lessons alone (logging on to the computer, opening files, saving files, logging off from the computer) took 15 minutes, so there was insufficient time for real programming and work. You can request teacher access at Scratch[2], which will allow you to set up a class and deposit materials.

<Cooperation activity>

Since the VPLS is an introduction to programming, there will be very few possibilities for cooperation.

As soon as the project is transferred to the real-life control of a microcontroller, cooperation between students would be useful, as the degree of difficulty of the problem increases by using additional components (sensors, pump). For example, older students could help to build the pump. Schools in different countries could work on the VPLS project together and compare their results and solutions.

A common database could be created for different plants to adapt the VPLS to the individual characteristics of a variety of plant species. If the VPLS is connected to the Internet, schools could adopt plants from another school and take charge of the watering.

<References>

[1] www.calliope.cc/en
[2] www.scratch.mit.edu/
[3] www.calliope.cc/en/los-geht-s/editor
[4] https://scratch.mit.edu/projects/editor/?tip_bar=getStarted (29/11/2018)
[5] All additional materials are available at www.science-on-stage.de/coding-materials.
[6] www.microbit.co.uk/home
[7] https://lab.open-roberta.org
[8] https://snap.berkeley.edu/

Magic Glove

<Author> Annamária Lőkös
<Author> Camelia Ioana Rațiu

<Info>

<Keywords> experiment, environment, temperature, humidity, luminosity, magnetic field

<Disciplines> physics, chemistry, biology, ecology, computer science

<Age level of the students> 10–18
<Level 1> for primary school (age: 10–11) and secondary school (middle school, age: 12–15)
<Level 2> for secondary school (high school, age: 15–18)

<Hardware> Arduino UNO[1], sensors compatible with Arduino (e.g. light sensor, temperature sensor, magnetic field sensor, humidity sensor, gas sensor), LCD button shield, jumper wires, external battery

<Language> C[2], Arduino 1.8.5[3], Snap![4]

<Programming level> medium

<Summary>

Young people are passionate about technology, so a lesson that combines science with computer science will be a successful one. The students will build and use a glove with a different sensor on each finger. This will allow them to carry out different experiments by connecting only the necessary sensor.

<Conceptual introduction>

The advantage of using a device (a glove) equipped with several sensors for different measurements is twofold. Primary and secondary school students can use the 'magic' glove to measure temperature, brightness, humidity, the presence of a magnetic field, sound intensity, etc. All they have to do is select the desired sensor and they are ready to start finding various uses for the glove in different fields of study and school subjects. The glove can be taken out in the field as it is powered by a battery; this offers a possibility for the students to investigate outside of the laboratory. On the other hand, high school students can build a glove themselves to make certain determinations. The students know the theoretical notions of the various sciences (physics, chemistry, biology, ecology), and they truly enjoy the opportunity to investigate them in practical experiments.

Teachers will need to present the basic concepts for coding a program in C[2] or any other programming language supported by Arduino, including Snap![4], for the students to code for Arduino[1]. To acquire a basic knowledge of C, if they choose it as programming language for Arduino, the students can watch tutorials on the Internet. This will help them to better understand how the code should be written and also boost their confidence since most of them will be surprised by just how simple the code is.

<What the students/teachers do>

<Level 1>

The glove has an LCD display with buttons. The students put on the glove and select the desired sensor with the **UP**/**DOWN** buttons; then they press the **SEL** button and the measurement starts. The screen displays the value. The students can repeat the measurements whenever they want. To return to the home screen, they press the **BACK** button. For example, you can see the determination of the poles on a magnet in 1a–1c.

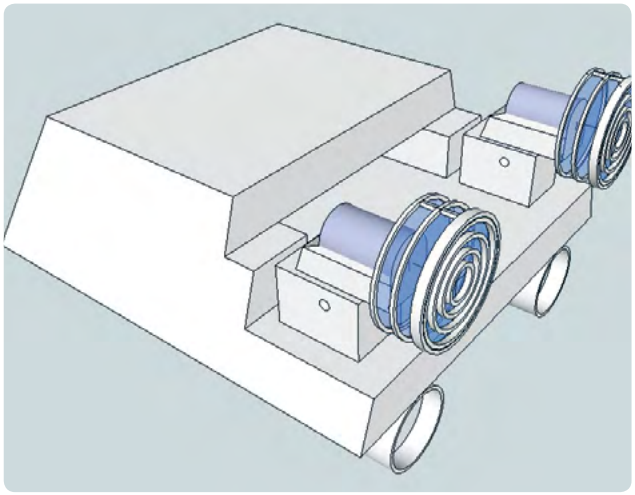
1a-c: Determining the poles of a magnet

<Level 2>

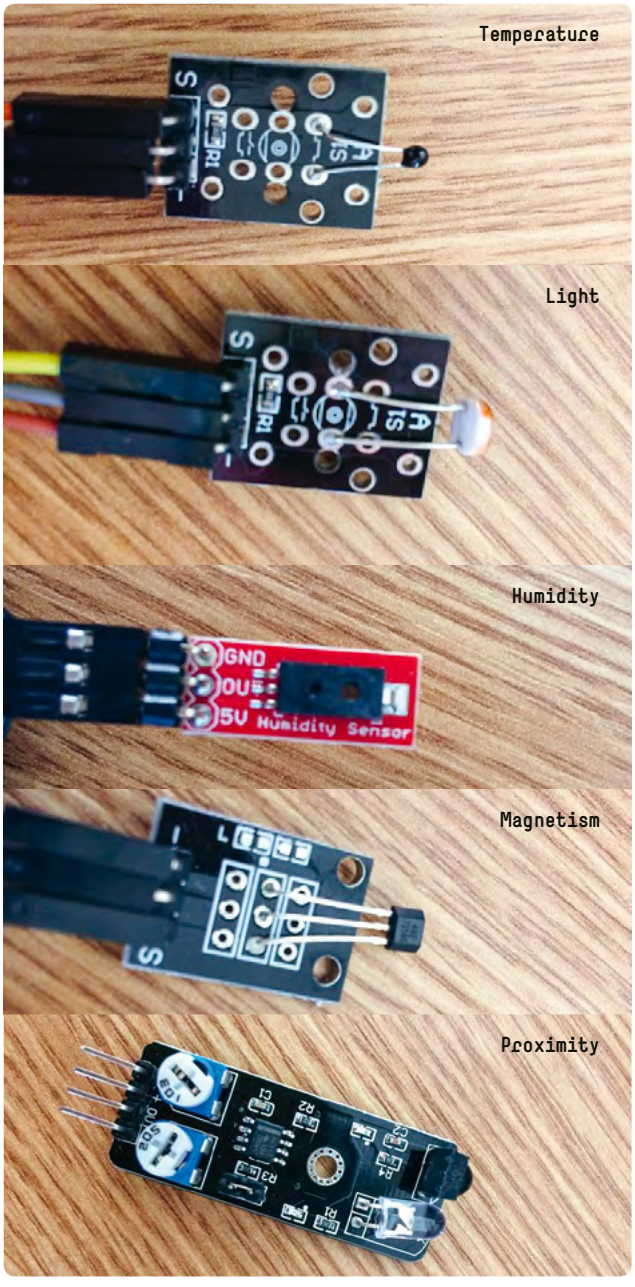
Students in a class can be divided into four groups. One group cuts and sews the glove, the second group makes the circuit, the third group does the coding and the last group calibrates the sensors.

<Making the glove>

The students made a template (2) after consulting documentation on several websites[5]. They folded the material (leather in our case, but other materials can be used) in three and cut it using the template. To get the proper glove, the students sewed two of the faces together. They stitched on the last piece of material after they had mounted the Arduino with LCD and sensors on the back of the palm. The students cut out the opening for the LCD display and buttons from this third piece.



2: Templates for the glove



3a-e: Sensors

<Building the circuit>

The students made the circuit, starting from a schematic diagram that they have discussed and analysed with the teacher beforehand. The circuit can be fixed (tinned), or not. It must take into account the correct connection of the sensors to the Arduino board, namely the **GND** from the sensor to the **GND** on the Arduino board, the **VCC** from the sensor to **5V** on the Arduino, and the **OUT** on the sensor to one of the **ANALOG IN** (A0, A1, A2, A3, A4 or A5) on the board. If a sensor has to be connected to the **DIGITAL IN**, care must be taken not to use one of the inputs used for the LCD because operating errors will occur. In our example, we connected the following sensors: temperature (A1), light (A2), humidity (A3), magnetism (A4) and proximity (A5) (see 3a–3e).[6]

<Writing the code for the circuit>

High school students who are studying the C programming language[2] can easily program Arduino[1]. There are many tutorials available online in a multitude of languages. For example, our students used a website in Romanian[7]. Tutorials in English are among others offered on the Arduino website or on the distributors' websites.[8] There are also many other sites where the students can find tutorials.

The teacher can guide them through how to write the program for Arduino, and you can find the complete code that we used online.[8]

<Calibrating sensors>

There are calibrated sensors available, but there are also uncalibrated ones; it is sometimes more enjoyable for the students to find a way to calibrate them. They found calibration formulas for some of the sensors on the Internet. For example, there is a formula for the humidity sensor brick[6], because the function by which the displayed values vary is not linear.

With regard to the calibration of the temperature sensor, the students tracked the values displayed by the sensor. They used a calibrated thermometer in the lab and associated the displayed value with the thermometer value. They discovered that this sensor varies linearly and found the calibration formula. There are examples with the calibration formulas for the humidity and temperature sensors in the additional material provided.[8]

Once the students have calibrated the sensors, completed the program and checked the display to ensure that the fingers of the glove are properly correlated with the data appearing on the screen, the glove is then mounted. The last step is to stitch/sew the outer layer of the material. The students used

some fastening rings for each sensor on each finger (📷4) to better fix the sensors.

📷 4: Fixing the sensors

<Algorithm to use in other languages>

If you would like to use another programming environment, a diagram with all the necessary elements for the main program is available online.[9]

<Conclusion>

Students enjoy discovering new things and are very inventive. They like to make experimental determinations, and the glove looks as though it came from a science fiction movie. High school students will enjoy the coding elements of the unit and see the results immediately in something practical that actually works.

This experience was an unprecedented one, and the teachers and students learnt many things together.

They did encounter one difficulty: it is not easy to find the right sensors[6] and calibrate them, but there are solutions. If a calibration formula cannot be found, a solution lies in purchasing calibrated sensors even though they are more expensive.

This glove could also be constructed using a Calliope mini, which would make it lighter and smaller. Our students are interested in trying to make such a glove even if the programming language is different.

<Cooperation activity>

Students from different schools and countries could make such gloves using various microcontrollers and suitable sensors, and then discuss and compare the results. An art teacher could be asked to contribute to the design of the glove. Additionally, a contest between schools could be organised, in which the students propose different designs themselves.

The glove is easy to send by mail, so students in different schools could experiment with the gloves made by their peers in other schools and other countries.

<References>

[1] www.arduino.cc

[2] https://en.wikipedia.org/wiki/C_(programming_language)

[3] www.arduino.cc/en/Main/Software

[4] https://snap.berkeley.edu/

[5] Several websites with documentation and tutorials on how to sew gloves: http://ofdreamsandseams.blogspot.ro/2012/04/1950s-hand-sewn-leather-gloves.html, https://so-sew-easy.com/easy-gloves-pattern-winter-comfort/, http://sew-ing.com/make/gloves.html, www.glove.org/Modern/myfirstgloves.php, www.instructables.com/id/How-to-Make-Gloves/ (all December 2018)

[6] We used the sensors in 'sensor kit 37 in 1' for Arduino. The humidity sensor is Humidity Sensor HIH-4030, brand: Sparkfun, code: SEN-VRM-09.

[7] www.robofun.ro (Tutorials in Romanian. Every product has instructions on how to connect it to the circuit and what programming language to use. The website contains diagrams and drawings, sensors or other components that can be recognised from the photos, and the program is very easy to follow. This means that the students do not need to have any knowledge of the Romanian language.)

[8] http://mthackathon.info/resources/37-SENSOR-KIT-TUTORIAL-FOR-UNO-AND-MEGA.pdf (tutorials are in English; they contain instructions on how to connect the sensors of the kit to Arduino) (December 2018)

[9] All additional materials are available at www.science-on-stage.de/coding-materials.

Science Friction

<Author> Ilia Mestvirishvili
<Author> David Shapakidze

<Info>

<Keywords> friction force, stopping distance, anti-lock braking system (ABS), app programming, data acquisition

<Disciplines> physics, computer science, mathematics

<Age level of the students> 14+

<Hardware> Arduino[1], servo, motor, Bluetooth module, motor shield, photogate

<Language> Arduino programming environment[2], AppInventor[3], Snap4Arduino[4], Blockly[5]

<Programming level> easy, medium

<Summary>

An investigation of the factors affecting the force of friction can be turned into an interesting and entertaining experiment by building a low-cost Bluetooth controlled car with a simple braking system. This will enable the students to observe real data such as the car's speed (absolute value of velocity) before applying the brakes, its stopping distance, and how the mass of a car and the type of surface affect friction force. The students will then conduct experiments to investigate the relationship between the factors affecting the stopping distance with sufficient precision to check their own hypotheses or those suggested by the teacher.

<Conceptual introduction>

Friction is a very important force in everyday life and is taught in physics at both middle school and high school level. However, traditional experiments related to the topic of friction are limited and not much fun. This project will turn the exploration of friction into an exciting group project which involves:

1. building and fine-tuning a car
2. programming an Arduino[1] microcontroller to measure instantaneous speed and stopping distance
3. programming a mobile phone using AppInventor[3] to send, receive and display real data on the phone screen

<What the students/teachers do>

Since these three tasks can initially be done simultaneously, we recommend that the teacher divides his/her class into groups of two to three students, who then work on the tasks separately, but come together to discuss and revise their work.

When friction is introduced in a particular curriculum, the teachers could offer the unit about friction in conjunction with this project, which will motivate the students and improve their understanding of the related theoretical concepts. The best way to start the project would be to pose the following questions to your students and give them time to brainstorm and come up with their own ideas, predictions or hypotheses:

- ↳ What is the relationship between the speed of a car and the stopping distance? (Answers might vary of course, i.e. 'the stopping distance is proportional to the speed before applying the brakes', or some might 'remember' from physics that the stopping distance is actually proportional to the square of the speed.)
- ↳ How would increasing the mass of a car affect the stopping distance, provided that everything else is unaffected? (A popular answer is that increasing the mass should increase the stopping distance.)
- ↳ How should we apply the brakes to stop a car as quickly as possible? (Possible answers: the best way is to stop the wheels completely; if we make them rotate in the opposite direction to the motion, this will stop the car more quickly; etc.)
- ↳ If both the front and rear brakes are identical, will they stop the car at the same time? (The students can reflect on their own experience with bicycles.)
- ↳ Any other questions that might come from the teacher or the students.

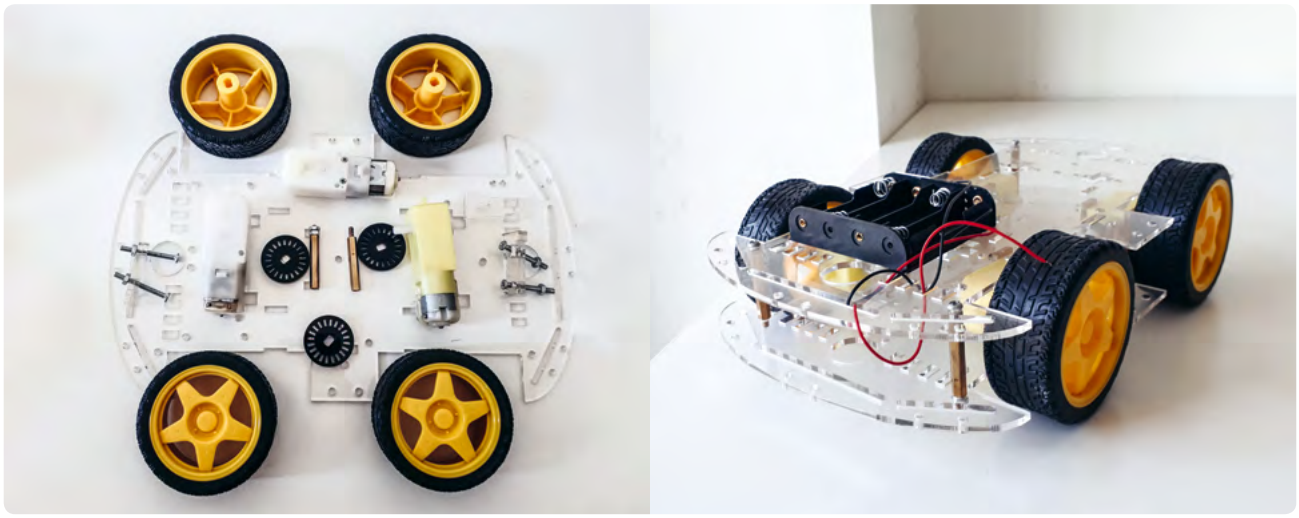
After the students have written down their initial ideas, the next step will be to think about how to build a simple car and the data that they will require to check and further develop their initial ideas. The teacher can facilitate this process and suggest that the students build a car that is able to collect and send relevant data to a phone, which in turn can control the car as well as receive and display the data.

Based on their interests, skills and preferences, at this stage of the project students can split into the previously mentioned groups if desired. However, just one group of students can do all these tasks as well. The next steps of the project development will be the same for either scenario.

<Building a chassis and mounting the electrical components>

This approach involves building a car from the low-cost and readily available Arduino[1] chassis kit depicted in 📷1. Teachers and students are encouraged to try different approaches to design and implementation, for example, different ways to collect and send data, and control the car remotely as well as different software and languages to write the necessary code.

After assembling the car and deciding where to mount the Arduino controller and the motor, the students will need to think of different ways to measure the speed of the car. The recommended approach is to brainstorm and give the students an



1: A complete chassis kit

opportunity to propose some ideas of their own. With appropriate help from the teacher, the best and easiest way to do this is to use a photogate to count the rotation rate of a free rear wheel. In this way, the students will be able to measure both the instantaneous velocity and the distance travelled. This can be done by just using the materials in the Arduino 4 wheel chassis kit, or developed separately if the teacher prefers to do so.

Some maths will be required here to transform the number of blocking events counted by the photogate into speed or distance. The kit includes a disc with 22 holes in it and a wheel with a diameter of 5.1±0.1 cm. It is not difficult to calculate that 1 pulse from the photogate, which means a wheel rotated 1/22 of a full rotation, corresponds to a distance of $d = 0.72$ cm. At the same time, the photogate measures and sends a time interval t in milliseconds between consecutive pulses. Instantaneous speed can be calculated by dividing 0.72 cm by this time interval.

The following steps can be used, irrespective of whether the students are working in different groups or only in one. A single group would work through all the steps one after the other, while different groups would split the three tasks.

<Arduino programming>

The Arduino programming group will work on the coding using the following approach:

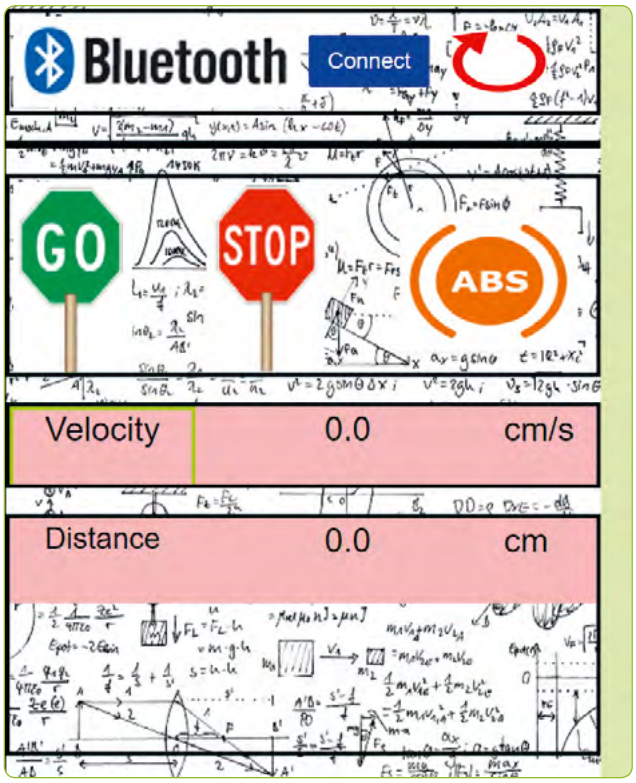
1. Define the actions and consequently the methods or functions to collect and send the required data via Bluetooth.
2. Write and test each method separately.
3. Put everything together.

Beginners could start with TinkerCad[6], which allows online Arduino circuit design and simulation, thus preventing burnouts and short circuit problems at the prototyping stage.

The following sections outline each part in more detail:

1. The required actions are: starting and stopping a motor, applying and releasing brakes, measuring the distance, measuring the speed, sending and receiving the data via Bluetooth.
2. The crucial part here is to write a code to measure the speed and distance travelled during the same experiment. Both of them use pulses from a photogate and are activated when '2' is received from the phone app via Bluetooth:
   - ↳ To measure the distance, there is a counter which starts counting pulses sent to the Arduino from a freely rotating rear wheel after the brakes are applied to the front wheels.
   - ↳ The time intervals between pulses are used to measure the instantaneous speed. A rear wheel turns 0.72 cm during one pulse, so this needs to be divided by the time interval between pulses.
   - ↳ The functionality of an anti-lock braking system (ABS) could be implemented by applying the brakes and releasing them once for a variety of time intervals between 50 and 200 ms (optimised experimentally), which in most cases leads to a shorter stopping distance.
3. When you put together a program for Arduino, you need to ensure that everything happens in one big loop. Therefore, if the program is interrupted at any particular step, it will affect all the following steps.

The sample code and references to other sources for each of these functionalities are available online[7] but, with a little help from their teachers, the students should be encouraged to try and write their own code.



2: User interface of the app

<Android programming>

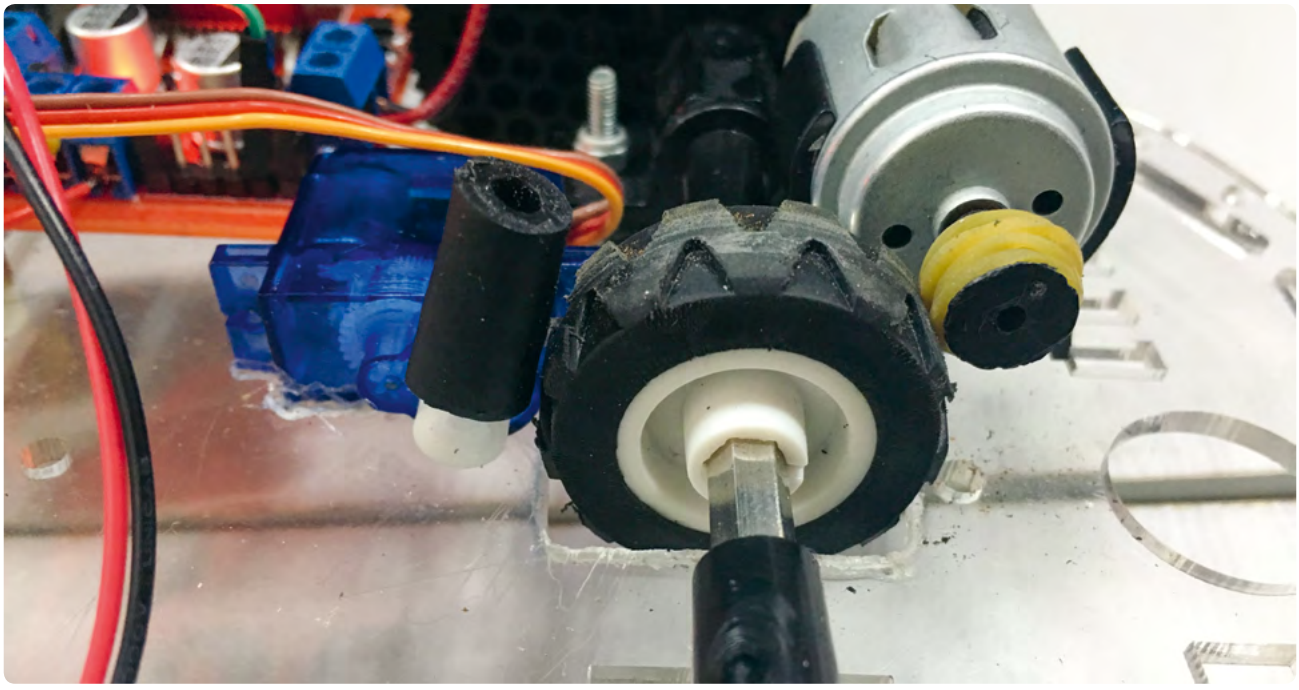
The Android programming group can explore AppInventor[3] and think of ways to display the data on the screen (user interface, UI). The students will decide where and how to arrange the buttons to control the car as well as the panels and labels to display the data on a screen (2). The code of the AppInventor program is provided online[7] and has the following functionalities:

1. Pushing the **START** button sends '1' to the Arduino[1] via Bluetooth and starts the motor in the car.
2. Pushing the **STOP** button sends '2' to the Arduino via Bluetooth, which stops the motor and then applies the brakes.
3. Pushing the **ABS** button sends '3' to the Arduino via Bluetooth, which stops the motor and then applies the brakes at regular intervals (simulating an ABS feature).
4. After pushing the **STOP** button or the **ABS** button, the data received about the instantaneous speed before stopping and the distance that the car has travelled after applying the brakes, i.e. the stopping distance, will be displayed in two corresponding panels with the 'speed' and 'distance' labels respectively.
5. The **RESET** button sends '0' to the Arduino via Bluetooth, clears the speed and distance data, and then resets the Arduino.

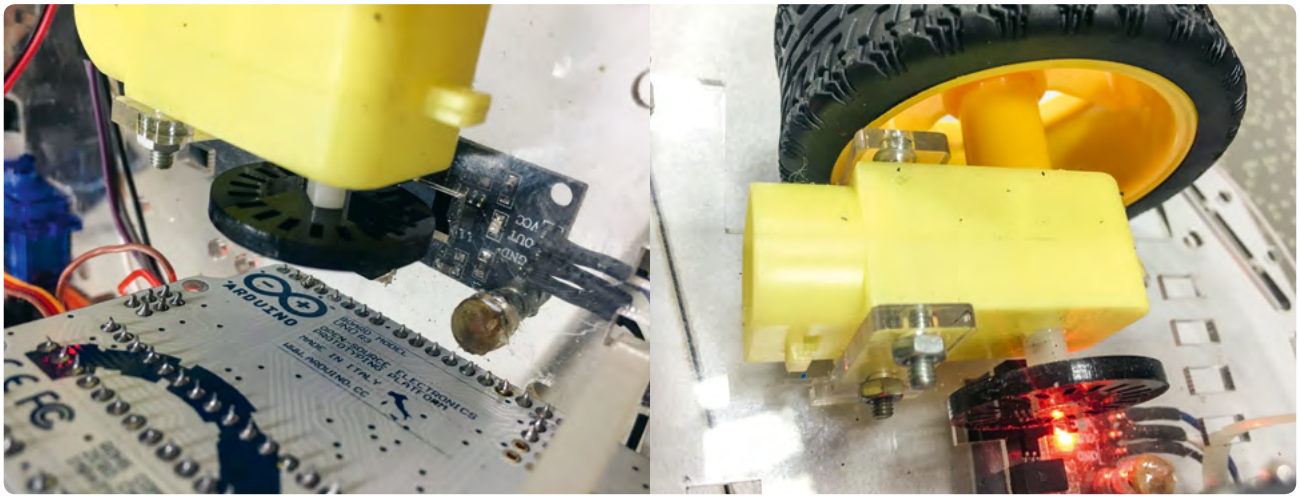
We recommend that you use the provided code as a reference for teachers and give the students the opportunity to explore AppInventor[3] and write their own code based on the above functionalities.

<Car building>

The car building group will be asked to think of appropriate locations and ways to attach components like the motor, photogate, servo, batteries, Bluetooth module, motor shield and, finally, the Arduino board itself. When the servo turns, it is important that it pushes a handle with a rubber tube firmly to the rotating disk, which is mounted on the front wheel axis (3).



3: A close-up of the brake system

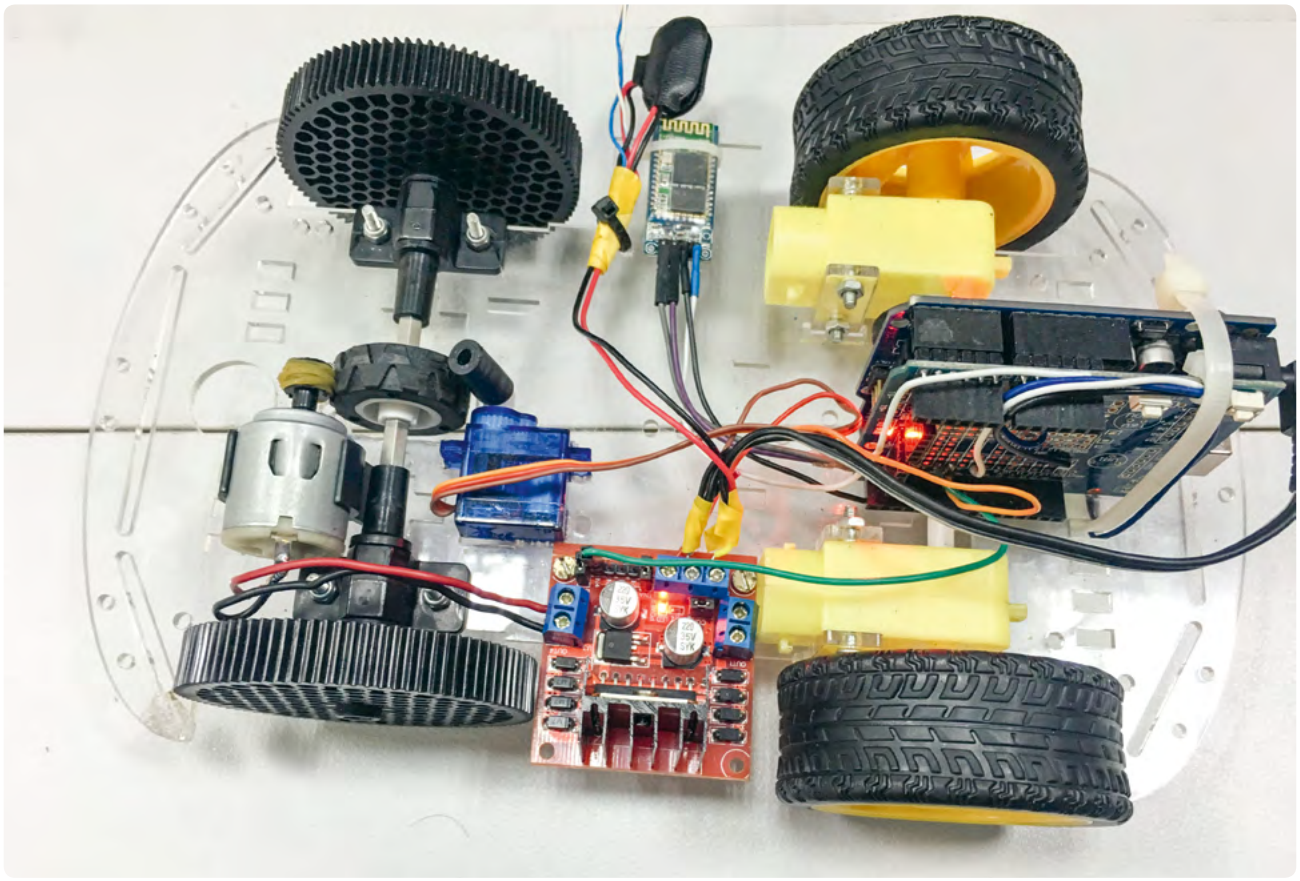


4: The photogate

It can provide enough force to stop the wheels immediately. The position of the photogate is equally important. Please ensure that it counts all the pulses correctly – the recommended photogate Arduino sensor has a built-in LED, which blinks when something enters the photogate gap. For this reason, please make sure that the photogate counts the rotations correctly when the rear wheels rotate (4).

Also, please ensure that the rear wheels rotate as freely as possible at all times. Remember that we calculate speed and distance using the free rotation of the rear wheels. Finally, the car should look similar to the example depicted in 5 once it has been built.

We recommend that you keep detailed guidelines as a reference and give students the opportunity to brainstorm and implement their own solutions.



5: The assembled car

<Conclusion>

This project is a fun way for students to learn core physics concepts like force of kinetic and static friction, and, at the same time, relatively modern technology like ABS, where physics, electronics, programming and design come together to explore factors affecting the stopping distance of a car. It is always a challenge to interpret and analyse real experimental data. It involves key concepts such as uncertainty, validity, reproducibility and visualisation. The project allows the students to experience and understand force of friction in a hands-on, thought-provoking lesson.

<Cooperation activity>

This project offers great potential for collaboration, since its three independent elements – the design of the car, the coding of the Arduino[1] and the coding with AppInventor[3] – could be further developed and improved. All the cooperating partners would benefit from each other's contribution to any of these components.

Another option for cooperation could be a competition between school teams based on who could make a car with the same wheels and the same mass stop more quickly, provided that the 'road' surface and speed before stopping are the same.

Many thanks to our Greek colleagues: Astrinos Tsoutsoudakis for providing important suggestions about the physics involved in this project and Georgios Georgoulakis for his extremely useful coding tips. We would also like to acknowledge the comprehensive feedback and support of Jörg Gutschank, which made this project more interesting and reproducible.

<References>

[1] www.arduino.cc
[2] www.arduino.cc/en/Guide/Environment
[3] http://appinventor.mit.edu
[4] http://snap4arduino.rocks/
[5] https://developers.google.com/blockly/
[6] www.tinkercad.com/circuits
[7] All additional materials are available at www.science-on-stage.de/coding-materials.

Rolling Sounds

<Author> Georgios Georgoulakis
<Author> Astrinos Tsoutsoudakis

<Info>

<Keywords> fundamental science, data acquisition, circular motion, sound waves, geometry, trigonometry

<Disciplines> physics, mathematics, computer science

<Age level of the students> 14–17

<Hardware> Arduino microcontroller[1] or similar (with the appropriate drivers installed), microphone, high output buzzer, power drill with its base, materials for wooden disk setup

<Language> Snap4Arduino[2]

<Programming level> medium

<Summary>

This interdisciplinary educational scenario combines physics with computer science. It can either be used in a computer science class or in a physics class and it involves—along with the calculation of other physical quantities of uniform circular motion—the calculation of the linear velocity of rotation in two different methods.

The first of these methods detects how often the signal of an infrared proximity sensor is blocked by a small metal strip that is attached to a particular point of a rotating disc, thus measuring the period. The second method exploits the Doppler shift of a sound-emitting source placed on top of the disc.

<Conceptual introduction>

The experimental study of the physical quantities (period T, frequency f, linear velocity v and angular velocity ω) of uniform circular motion is based on the knowledge acquired in Greek senior secondary schools (students' age: 14–17) and in the curricula of other European secondary schools. This also involves learning about the Doppler effect. The frequency and the magnitudes of angular and linear velocity are obtained by using well-known formulas:

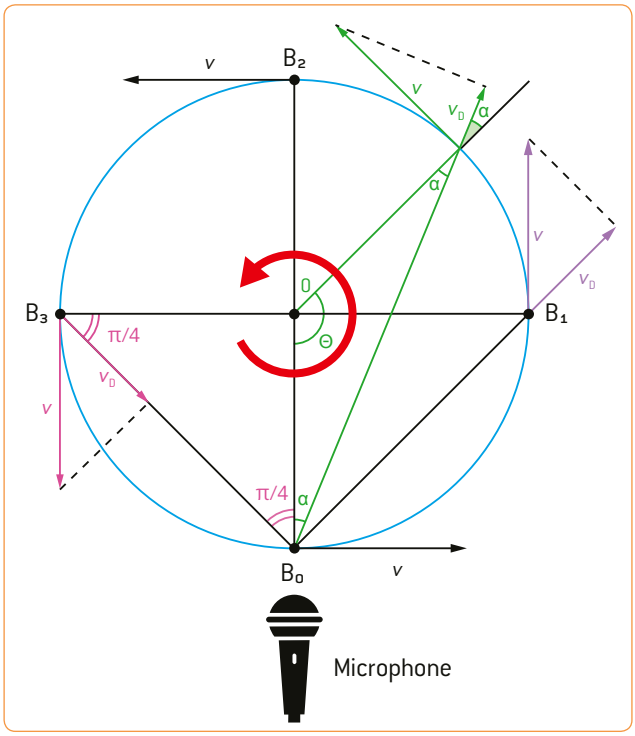
$$f = \frac{1}{T},\ \omega = \frac{2\pi}{T} \text{ and } v = \frac{2\pi r}{T}$$

T is derived from the microprocessor's internal clock, i.e. the time between signal detections, and radius r stands for the distance between the metal strip, or the buzzer, and the centre of the disc.

<Doppler effect experiment>

The Doppler effect is the change in frequency or wavelength of a wave if the source moves relatively to an observer. A common day-to-day example of this phenomenon is the change in pitch of a siren on a moving ambulance. When the ambulance approaches, the sound heard is higher than the original one; however, when the ambulance moves away, it is lower. The received sound is only the same at the moment when the ambulance passes the observer.

We use a buzzer on a rotating disc as the source and a static microphone as the observer in our experiment (see 1).



1: The schematic of the experiment

As the disc rotates counter-clockwise (1), the component of the velocity on the line of the chord, joining point B_0 with the point where the buzzer B is located every time, increases from zero to maximum at point B_1 and subsequently decreases to zero at point B_2. That velocity component is the actual receding velocity for the Doppler effect. From point B_2 to B_3, the velocity component, which now stands for the approaching velocity, increases from zero to maximum at point B_3 and then decreases again to zero at point B_0.

Calculate the linear velocity by applying the Doppler shift formula for a moving source at point B_3 and an observer at rest. The linear velocity remains constantly perpendicular to the radius of the circle and the angle of $\frac{\pi}{4}$ is determined by the geometrical properties of the formed right-angle and isosceles triangle B_3OB_0.

$$f = f_0 \cdot \left(\frac{v_s}{v_s - v_D}\right) \Rightarrow f = \frac{f_0 \cdot v_s}{v_s - v_D} \Rightarrow f \cdot v_s - f \cdot v_D = f_0 \cdot v_s$$
$$\Rightarrow f \cdot v_D = (f - f_0) \cdot v_s \Rightarrow v \cdot \cos\left(\frac{\pi}{4}\right) = \left(\frac{f - f_0}{f}\right) v_s$$
$$\Rightarrow v = \left(\frac{f - f_0}{f}\right) \frac{v_s}{\cos\left(\frac{\pi}{4}\right)}$$

f: measured frequency
f_0: emitted frequency
v: velocity
v_s: velocity of sound
v_D: receding/approaching velocity

As the implementation of a Fast Fourier Transform to extract the frequency content of the produced sound is well beyond the students' coding capabilities, free audio editing software like Audacity[3] will provide an adequate text file with all the necessary data.

<Further materials>

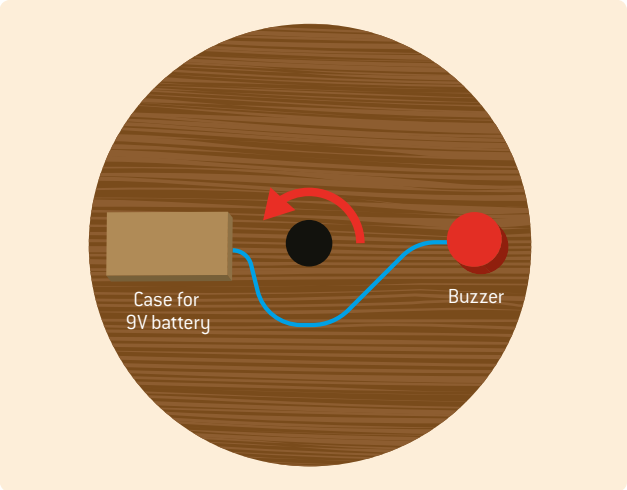
One more method that utilises a pitot-static tube system and a differential pressure sensor has been deliberately left out of the scope of this teaching unit to simplify matters, but all the required information is available online[4]. The online material offers a detailed description of the experimental setup, alternative construction ideas, along with theoretical documentation and step-by-step analysis of the procedures used.

<What the students/teachers do>

In the physics-related section of the teaching unit, the students will measure the physical quantities of circular motion at different radii and explore the Doppler effect. However, they will first design and assemble a basic experimental setup beforehand.

<The wooden disc setup>

The students will build the setup consisting of a wooden disc which is driven by a power drill and bears a buzzer connected to a 9V battery. An independent but closely situated infrared proximity sensor feeds the microcontroller a signal for every complete rotation, while a cheap microphone records the sound that is produced. The Doppler shift should ideally be audible for the chosen rotational speed, which should be kept low for safety reasons. (📷2 & 3)



📷 3: Disc viewed from above

📷 2: Basic experimental setup

An attractive and student-friendly interface has been developed as shown in 4 to input all the required parameters and extract the calculated values.

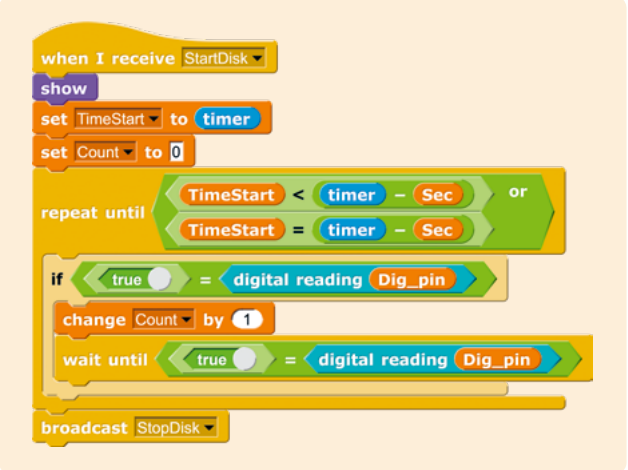
4: The experiment's interface

The students need basic programming skills and some experience using block programming languages (such as Scratch or Snap!). We give the students a basic template to work their code to ensure that they focus on the teaching unit objectives and not on the user interface and the appearance of the program.

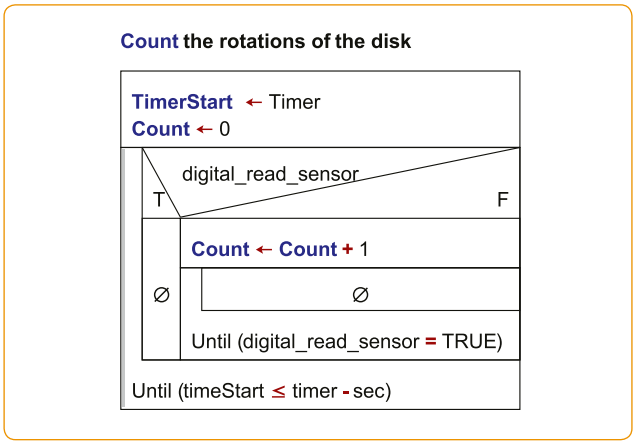
Therefore, we provide the basic template in a Snap![5] project .xml file and a work assignment paper to give the students basic instructions about the template form and outline what we expect from them. Both the template and the assignments are available online.[4]

The students will check and validate the acquired data, connect and communicate with the external devices, receive and process data from the sensors and write a simple serial search algorithm.

The finished program is also available for the teacher for download as a reference file[4]. 5 contains an example screenshot of the Snap4Arduino[2] programming environment. 6 contains the corresponding Nassi-Shneiderman diagram.



5: The calculation of the rotational period

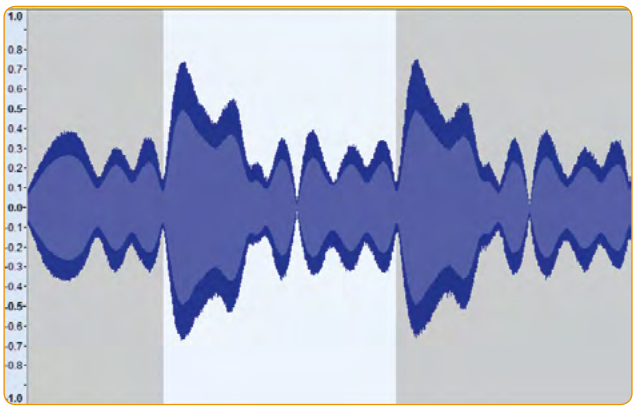


6: Nassi-Shneiderman diagram for period

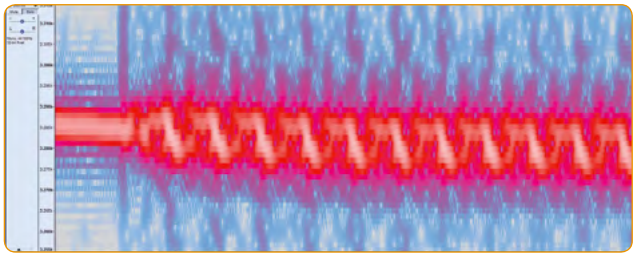
As mentioned before, we recommend using a free audio editing software like Audacity[3] to extract the frequency content of the produced sound. The software provides an adequate text file with all the necessary data for the students to learn how to process a sound signal by means of a specialised software.

The signal import and essential processing is illustrated in 7–9, while 10 depicts a part of the final data export. To avoid in-depth analysis and to allow for a better comprehension from the students, a rough assumption has to be made here. The yellow highlighted frequency, which possesses the maximum level, is the buzzer's frequency at rest or the one measured at points B_0 and B_2, while blue and green represent those between our primary frequency shifts as the nearest peaks (10).

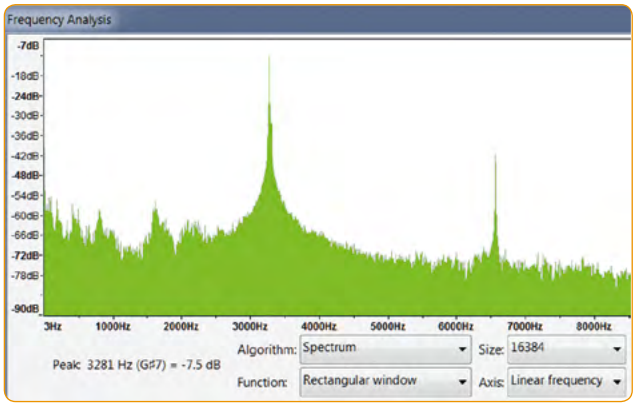
However, the proposed code searches only for the green highlighted, but for better accuracy it can be easily modified to find both. To speed things up, the data section could also be narrowed to only 50–100 values above and below the frequency of the maximum level.



7: A recorded sound waveform

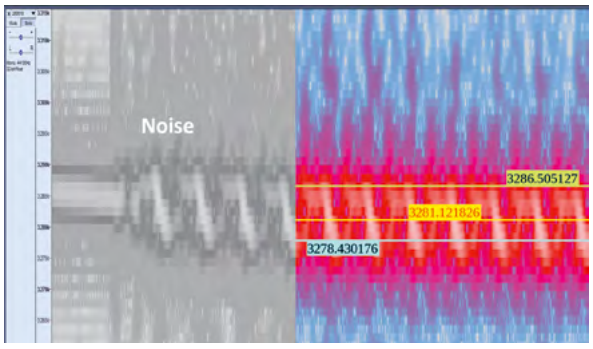


8: A spectrogram showing the Doppler shift



9: Frequency analysis by Fast Fourier Transform

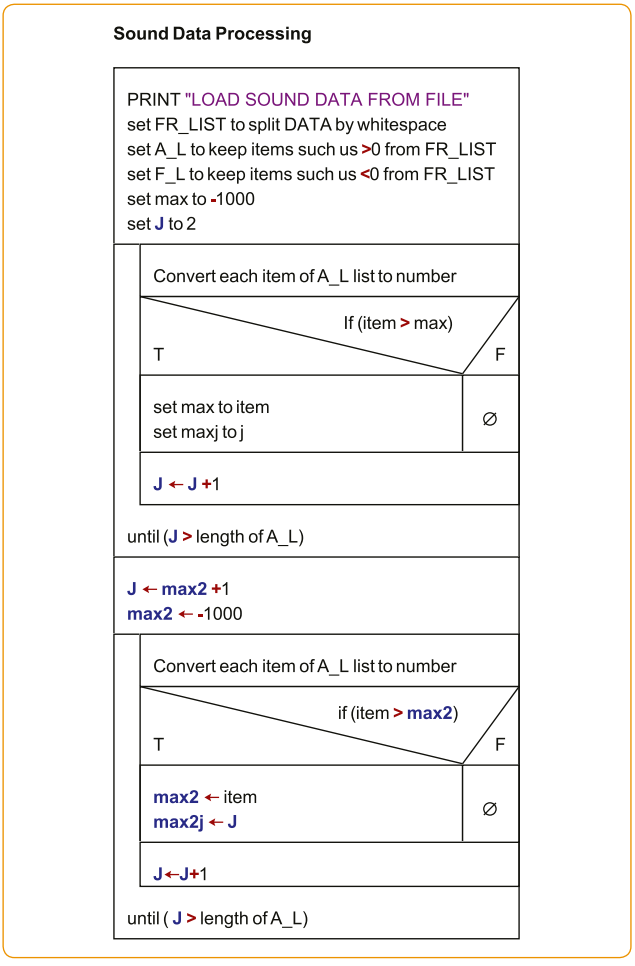
Frequency(Hz)	Level (dB)
3273.046875	-27.597595
3275.738525	-22.331339
3278.430176	-12.437067
3281.121826	-7.5547090
3283.813477	-10.041918
3286.505127	-9.7750780
3289.196777	-16.848948
3291.888428	-26.916197



10: Exported data

The sound spectrum data processing is shown in 11 & 12. Detailed information, e.g. about the used variables, is available online. [4]

11: The Doppler shift part of the experiment[4]



12: Nassi-Shneiderman diagram for sound data processing

<Algorithm to use in other languages>

The basic template will allow easy transfer to any other programming language as long as there is a basic library for communication with the microcontroller. Hence, the choice of microcontroller will have no significant effect on the project.

<Conclusion>

This is a low-cost project which is easy to assemble and operate; it will hopefully be interesting and stimulating for the students.

<Cooperation activity>

The Science on Stage platform is all about exchanging teaching ideas and implementing innovative educational approaches. Co-teaching with Ilia Mestvirishvili and David Shapakidze, a superb partner team from Georgia, might prove to be challenging due to the distance and scheduling issues, but it has already allowed us to develop new techniques. Despite the common lack of a background in special needs education, it would be a good idea to modify the project to make it accessible to all students.

<References>

[1] www.arduino.cc
[2] http://snap4arduino.rocks
[3] www.audacityteam.org
[4] All additional materials are available at www.science-on-stage.de/coding-materials.
[5] https://snap.berkeley.edu

↳ www.physicsclassroom.com/mmedia/circmot/ucm.cfm
↳ https://education.pasco.com/epub/PhysicsNGSS/BookInd-904.html
↳ http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/dopp.html
↳ http://newton.phys.uaic.ro/data/pdf/Doppler_experiment.pdf
↳ https://manual.audacityteam.org/man/tutorials.html
(all December 2018)

Physics Engine

<Author> Mihaela Irina Giurgea
<Author> Corina Lavinia Toma

<Info>

<Keywords> animation, sprite, blocks, loops, graphs, gravitational laws, collisions, free fall, friction force, oblique throwing, motion, momentum, operators, variables

<Disciplines> computer science, physics, mathematics, ICT

<Age level of the students> 14–16

<Hardware> computer

<Language> Scratch[1]

<Programming level> easy, medium

<Summary>

What would you think if we told you that your students could learn two seemingly very different subjects, physics and computer science, more easily and also at the same time? In this unit, the 'engine', i.e. the Scratch[1] programming environment, is the magic tool that will help students to create interesting applications about everyday natural phenomena to better understand the laws of physics and improve their programming skills in the process.

<Conceptual introduction>

Why did we use Scratch[1]? Scratch is a visual programming environment that animates sprites using blocks on the computer screen and helps students to create applications more easily than with classical programming environments (C++, Java, etc.). In addition, our 'engine' helps us to teach two subjects that students consider difficult: physics and computer science. By removing the main obstacles, the impossibility to imagine (see) how a phenomenon actually works and the complicated syntax of coding, we have created an enjoyable new way of teaching.

The students who were involved in the development of this teaching unit already had some coding experience, so they were able to work out how to use Scratch. They use it in their regular as well as in their optional computer science lessons. Besides this, the physics knowledge required is part of the standard curriculum, and it is always useful to revise and apply what you have learned.

<What the students/teachers do>

The unit is all about alternating sequences of learning involving coding and physics.

First, the computer science teacher presented the basics for making a project in Scratch[1]. The students familiarised themselves with several key words related to the Scratch environment: stage, sprites, costumes and movement. You can follow the instructions and explanations for the first application taught in Scratch, an application without physics formulas.[2]

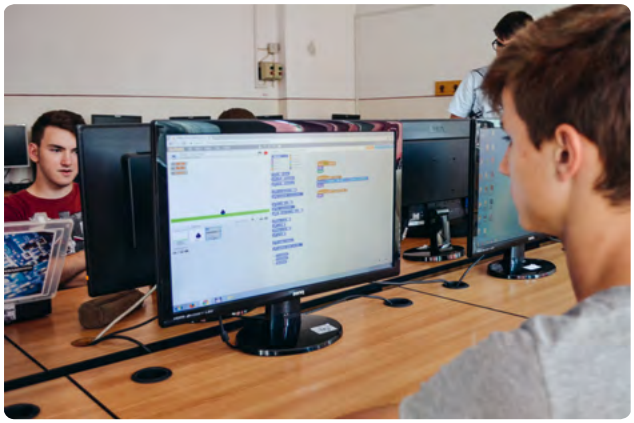
The students needed to understand the interactions between the sprites and their synchronisation as well as how a coordinate system works. You can find a complete tutorial for Scratch online.[3]

To better understand the main algorithms in Scratch, the students looked at exciting and interesting apps. When they saw the code behind these, they were sometimes surprised to discover that they could create such apps themselves.

For the physics part of this unit, the students applied the theoretical principles behind the phenomena of the world around them. For this reason, the physics teacher suggested a wide variety of topics[4], which the students then discussed: formulas needed, possible animation, the design, etc.

The students chose from those topics. A week later, we received apps on the oblique throw of a ball, the free fall of an apple, the more complex fall of a water drop or the collision between two balls, but also the movement of planets in the Solar System or even small games.

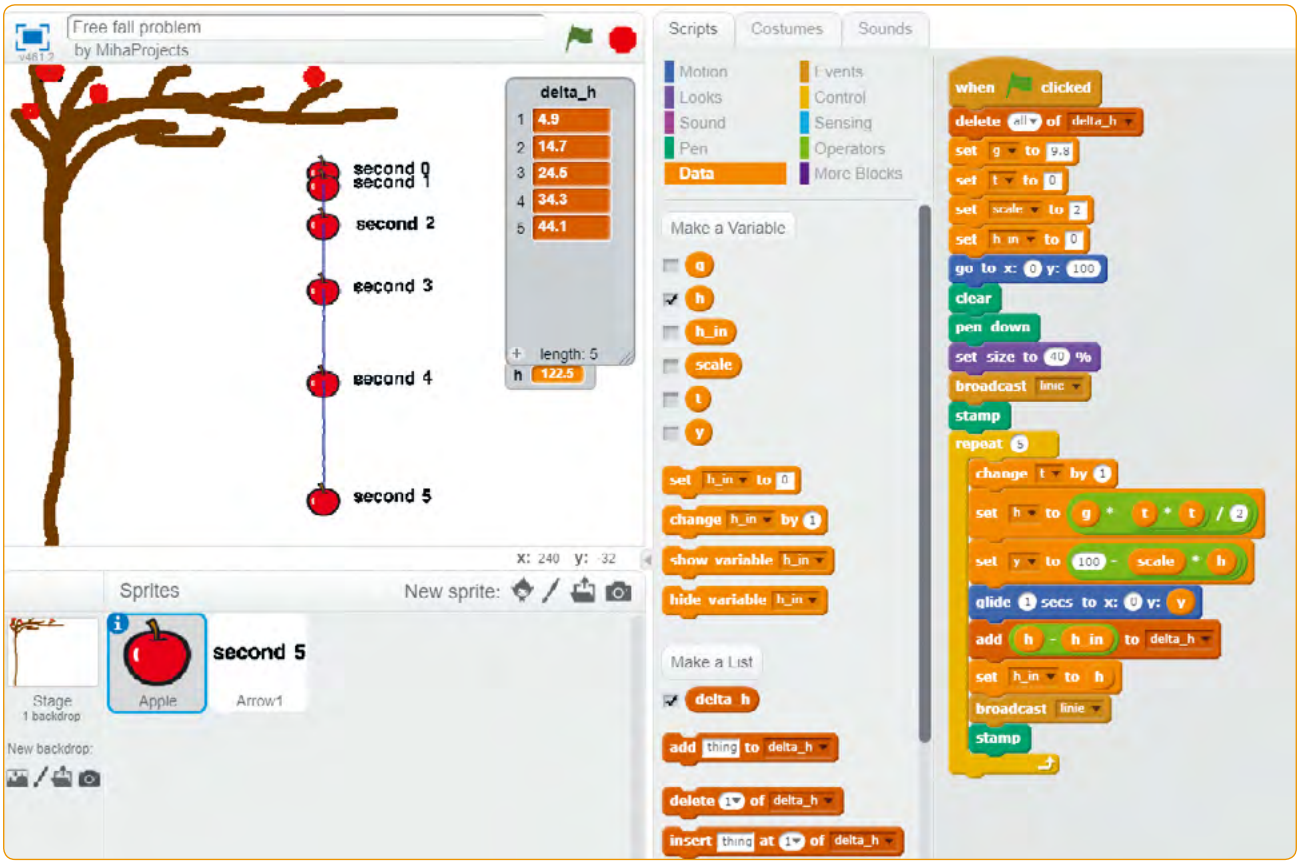
At first, the students worked individually with some help from their teachers. When the projects needed improvement, the students were guided by the teachers and their classmates. (📷1)



📷 1: Individual work

Then, everyone presented their project to the class and received feedback from their peers. This made it easier for the students to work out which parts of the projects needed to be improved: the coding or the physics.

At the end of our project, the older students became teachers for the younger students (12–13 years old) by presenting



2: Free fall problem

suitable applications and testing them during physics lessons. They enjoyed the responsibility and were very proud of their work. The older students also received suggestions from the younger students. The best of these student simulations are available on the Scratch platform.[2]

The following sections contain examples of how we approached the physics and the programming parts.

<Application 1: Free fall problem (Newton's apple)>

Every student has heard of the free fall of this historical object: Newton's apple.

The application in 2 is inspired by a classical problem: what is the distance travelled every second by Newton's apple in free fall?

Physics theory

We consider a linear motion with constant gravitational acceleration $g = 9.8 \frac{\text{m}}{\text{s}^2}$.

After a time t, the travelled distance $h(t)$ of the apple is: $h(t) = \frac{gt^2}{2}$.

The initial point $h(0)$ is fixed on the tree branch from which the apple detached itself.

Then we calculate the travelled distance during a longer period, $t + \Delta t$: $h(t+\Delta t) = \frac{g(t+\Delta t)^2}{2}$.

The general formula for the distance $\Delta h(t)$, the distance travelled by the apple during Δt, is:
$\Delta h(t) = h(t+\Delta t) - h(t) = \frac{g(2t\Delta t + \Delta t^2)}{2}$.

Then we use the data of the specific problem to customise the general formula; in this case, 1s for Δt. For the first second, $t = t_{\text{in}} = 0$ results in $\Delta h_1 = 4.9$ m, for the next second, t=1s results in $\Delta h_2 = 3 \cdot 4.9$ m $= 14.7$ m and so on. Through mathematical induction, we can calculate the distance travelled during the n^{th} second considering $t = (n-1)$ s:
$\Delta h_n = \frac{9.8(2n-1)}{2}$ m.

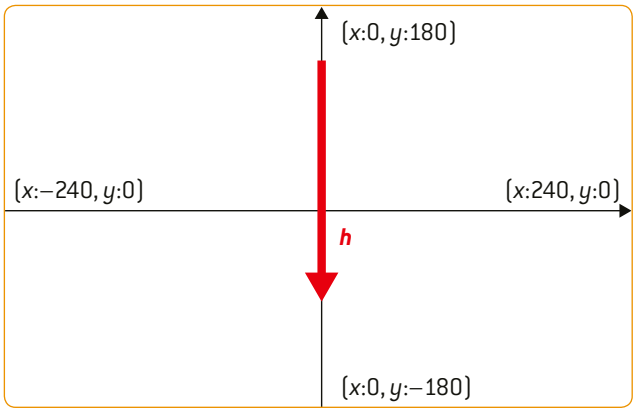
Then the students can calculate, and also see in our animation, that the travelled distance increases by the same amount every second, 9.8 m.

How do we code this?

Variables used:
g: gravitational acceleration
t: a counter for seconds (with values: 0, 1, 2, 3, 4, 5)

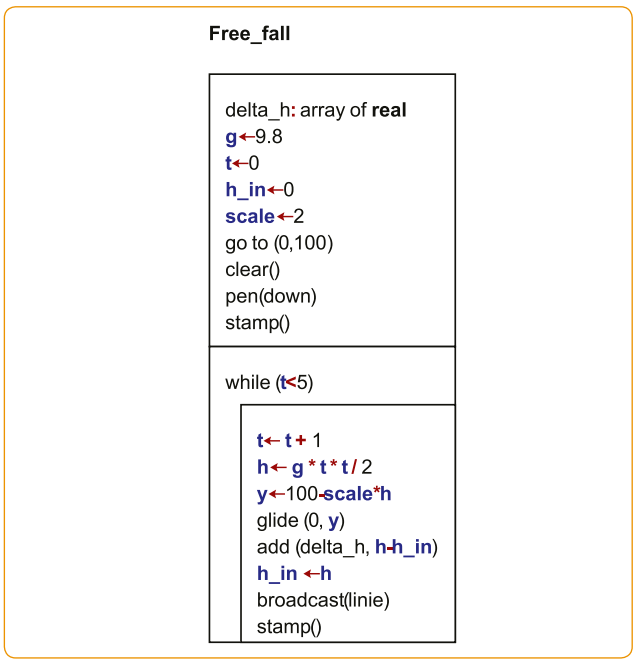
h: the travelled distance after *t* seconds
h_in: the initial position of the apple
delta_h: a list (array) with all distances travelled in every second
y: the *y*-coordinate of the apple
Observation: in this app, the *x*-coordinate remains constant = 0, so you can move the trajectory more easily to the left or the right on the stage.

At the beginning, the apple is at the point with the coordinates (0,180). The initial point and the direction for the travelled distance *h*(*t*) are marked in 3.



3: Orientation in the coordinate system

Using a loop 5 times, we recalculated the distance that the apple travelled after every second, working out the new *y*-coordinate and considering the screen's characteristics. See 4 for the encoding algorithm.



4: Free fall problem

Challenge

The students modify Δ*t* and the travelled time *t* (our apple tree should be very tall—it is probably better to draw a tower building) or move the problem to another planet with its own gravitational acceleration. To create a complex project, they could add the friction force from the air and consider a variable gravitational acceleration by dropping the apple from a weather balloon at a higher altitude.

<Application 2: Falling water drop>

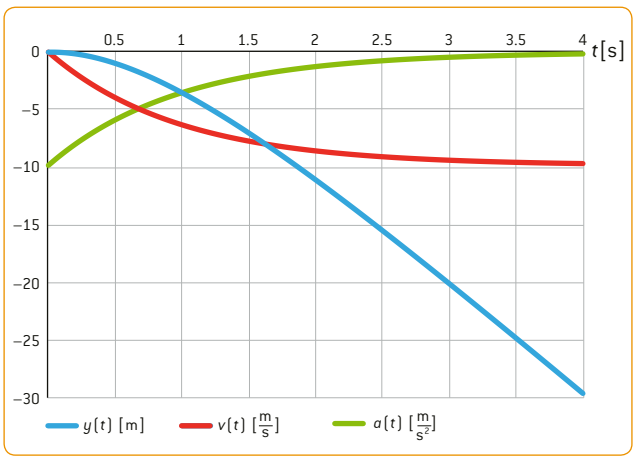
On a rainy day, everyone can observe the fall of water drops. The students analysed the linear movement of one drop with our simulation. At the beginning, they saw that the fall of the water drop is accelerated, but with decreasing acceleration. After some time, the drop's velocity reached its limit, the terminal velocity v_t, when the acceleration reached zero. Then the water drop continued to move at this constant velocity. How can you explain this?

Physics theory

In the accelerated part of the movement, two forces act in opposite directions on the drop: the gravitational force $G = mg$ (*m*: mass of the drop, *g*: gravitational acceleration) and the friction force $F_f = kv$ (*k*: constant of proportionality, *v*: instantaneous velocity). The acceleration of the drop becomes: $a = g - \frac{k}{m} v$.

In our simulation, we consider a large drop that measures about 5 mm in diameter with the terminal velocity $v_t = 9.8 \frac{m}{s}$.[5] In this case, the constant $\frac{k}{m} = \frac{1}{s}$. The acceleration decreases when the velocity increases (see 5). The initial values are $a = 9.8 \frac{m}{s^2}$, $v = 0$ and $y = 0$.

We use a small program in C++[4] to calculate the instantaneous acceleration and velocity, where we consider the acceleration and the velocity constant for very small time intervals Δ*t* (like 0.05 s). In this case, the velocity increases with $\Delta v = a\Delta t$ and the travelled distance with $\Delta y = v\Delta t$ for each chosen Δ*t* (step-by-step method).

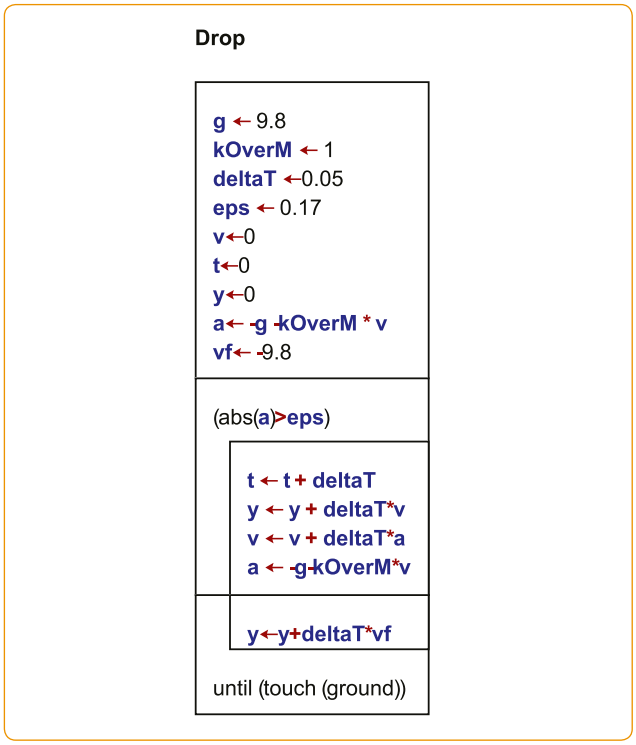


5: Relation between travelled distance, velocity and acceleration

How do we code this?

1. Paint a water drop sprite.
2. Paint a horizontal green line at the bottom of the backdrop.
3. Make a sprite with the message 'acceleration=0', which appears when the acceleration has a value of about 0.
4. Write the code for the drop sprite. The drop starts at the point $(0, y_{init})$. Using a loop, we recalculate $a(t), v(t), y(t)$. We use the distance reached by the drop after every Δt, working out the new y-coordinate and taking into account the screen's characteristics. The acceleration decreases and when it is about 0, the loop is finished. At this point, the sprite message is shown on the screen. Next, the drop sprite falls at a constant velocity until it touches the green line of the backdrop.

📷6 provides a clear explanation of the code.



📷 6: Falling drop

Challenge

The students can improve this application if they add a variable for the water drop mass (the water drop diameter can usually take on values from 1 mm to 5 mm)[6] and another type of friction force: $F_f = \frac{kv^2}{2}$.

The students could also add more drops of different masses and compare how they fall.

<Application 3: Elastic collision>

There are many examples of bodies colliding around us. These collisions are complicated, but we consider the elastic collisions with applicability in real life for billiard or steel balls, or in theory for collisions of molecules when the students study the ideal gas model.

Physics theory

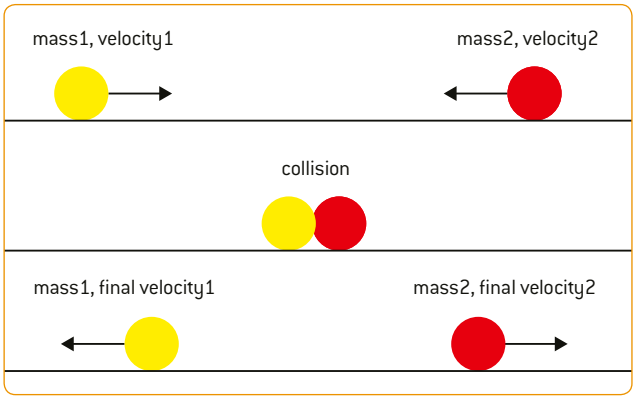
The linear momentum and the kinetic energy are conserved for two balls with the mass m_1 and m_2, the initial velocities $(\vec{v_1})$ and $(\vec{v_2})$ and the final velocities $(\vec{v_{1f}})$and $(\vec{v_{2f}})$. [📷7]

$$m_1\vec{v_1} + m_2\vec{v_2} = m_1\vec{v_{1f}} + m_2\vec{v_{2f}} \text{ and}$$

$$\frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 = \frac{1}{2}m_1v_{1f}^2 + \frac{1}{2}m_2v_{2f}^2$$

If all motion takes place along the same line (movement on x-axis), we can use + or – signs to designate directions. Vector notation is not needed for the straight-line collision case, and the final velocities can be calculated with the following equations:

$$v_{1f} = 2\frac{m_1v_1+m_2v_2}{m_1+m_2} - v_1 \text{ and } v_{2f} = 2\frac{m_1v_1+m_2v_2}{m_1+m_2} - v_2.$$



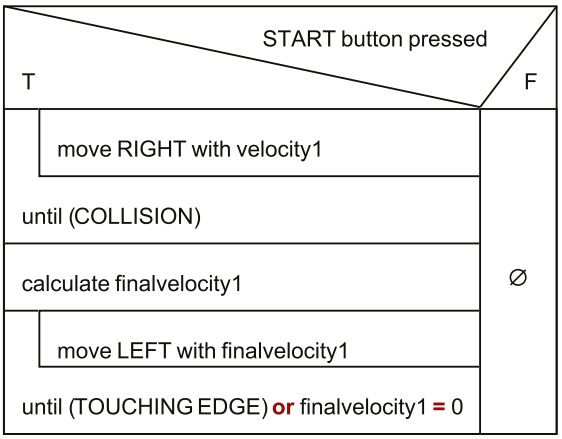
📷 7: Elastic collision

How do we code this?

1. Choose two sprites for the balls (Ball1 and Ball2) and one sprite for the START button (Start sprite).
2. Use variables: *mass1*, *mass2*, *velocity1*, *velocity2* (the mass and the initial velocity) for each object. Make the variable sliders visible and set the minimum and maximum value for them.
3. Enter the mass and the initial velocities for each object.
4. Press the START button. At this time, the sprite broadcasts a message for the ball sprites. When they receive the message, each ball moves towards the other using the well-known formula *distance = speed × time*.
5. Calculate the final velocities of the balls and use them to move the balls in the right direction until a ball either touches the edge and leaves the scene or stays in its place because its new velocity is 0.

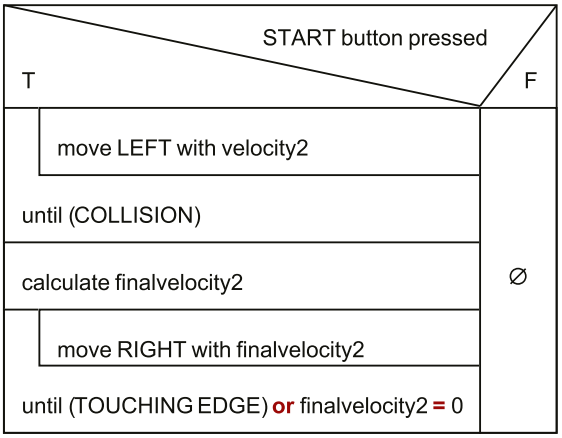
📷8 and 9 provide a clear overview of how the two balls are animated in Scratch[1].

Elastic_Collision_Ball1



8: Elastic collision for Ball1

Elastic_Collision_Ball2



9: Elastic collision for Ball2

Two examples of using this application:

1. Choose one velocity 0 and equal mass for the balls; after this collision, you will observe that the moving ball stops and the other one moves with the same velocity that the first ball had before the impact.
2. The balls have different velocities and equal mass; after the collision, you will see that the objects take each other's value for the velocity.

In both examples, the balls interchange their momentum.

Challenge

The students could change the size of the balls directly proportional to their mass; they could make an application for a two-dimensional elastic collision (simulation for the Compton effect) or they could program a simulation for the collision of a ball with a wall (reflection law in mechanics).

You could continue the study with another application, i.e. an inelastic collision.[4]

<Conclusion>

<For the students>

Advantages

The students learned physics theory in a more enjoyable way and were able to understand the natural phenomena better using simulations in Scratch. They deepened their computer science and physics knowledge at the same time. Even though their projects were not all perfect, the students clearly improved their coding and algorithmic thinking skills as a result.

Disadvantages

The students worked alone and more at home. They received feedback at school.

<For the teachers>

Advantages

We observed a real interest in creating an original application and learning more than in classical lessons.

Disadvantages

It was difficult for us to coordinate the whole class because of the wide variety of physics topics and the very specific bugs in each application. We think that it would be better to give all the students the same topic and to encourage them to improve it to the best of their varying levels of ability.

<Cooperation activity>

Students from different schools and countries could solve the challenges of the projects and create new ones with other ideas related to the original topic. All these applications could be put in the same place on the Scratch platform, and then a contest could be organised to determine the best of them. Teachers also need to take into account the complexity of the coding and the physics when they evaluate their students' work.

<References>

[1] https://scratch.mit.edu/
[2] All additional materials are available at www.science-on-stage.de/coding-materials.
[3] https://en.scratch-wiki.info/
[4] https://scratch.mit.edu/users/SonS_Coding
[5] http://hypertextbook.com/facts/2007/EvanKaplan.shtml (29/11/2018)
[6] https://journals.ametsoc.org/doi/pdf/10.1175/1520-0450%281969%29008%3C0249%3ATVORA%3E2.0.CO%3B2 (29/11/2018)

SMB-Science Magic Box

<Author> Luc Ivarra
<Author> Marco Nicolini

<Info>

<Keywords> microcontroller, transducer, sensor, actuator, signal, physical quantity, loop, branch, sequential, process, calibration, input, output, read, write, analogue, digital, linearity, conversion, breadboard, pin, soldering, human machine interface

<Disciplines> physics, electronics, mathematics, ICT, logic, biology

<Age level of the students> 14–18

<Hardware> Arduino UNO[1] with Arduino DUE[2] and/or TI-Nspire CX CAS with TI-Innovator Hub

<Language> C++ (using Arduino IDE[3]) and/or TI-Basic

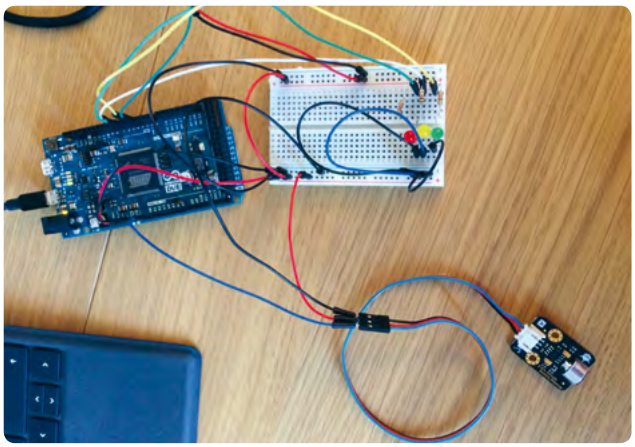
<Programming level> medium, with an audio section for advanced students

A list of the abbreviations, special terms and acronyms used in this unit is available online.[4]

<Summary>

The students will learn how to program for a self-built hardware-software environment (based on Arduino) and for a ready-to-use pocket computer (TI-Nspire CX CAS calculator with its extension, the TI-Innovator Hub). Both are used as devices for sensor data collection, conversion and transduction to easily handle, read, convert and actuate physical quantities.

Using the Arduino platform, the students will code in a framework of sensors that acquire physical quantities as input signals, and actuators that react to the acquisition and produce an output signal as a physical quantity after a microcontroller has processed the detected signal to set the proper output (see 1).



1: Arduino board

The TI-Innovator Hub is a 'ready to use' box that enables students to learn the basics of programming. It must be plugged into a TI-Nspire CX CAS calculator. It has a good I/O interface, which includes a luminosity sensor, two LEDs and an on-board buzzer, which produces a sound of a given frequency (see 2).

2: TI-Nspire and TI-Innovator Hub

<Conceptual introduction>

The unit introduces students to the coding world for physical problems, based on detecting and measuring physical quantities, processing the data, reacting and deciding to perform an action with actuators.

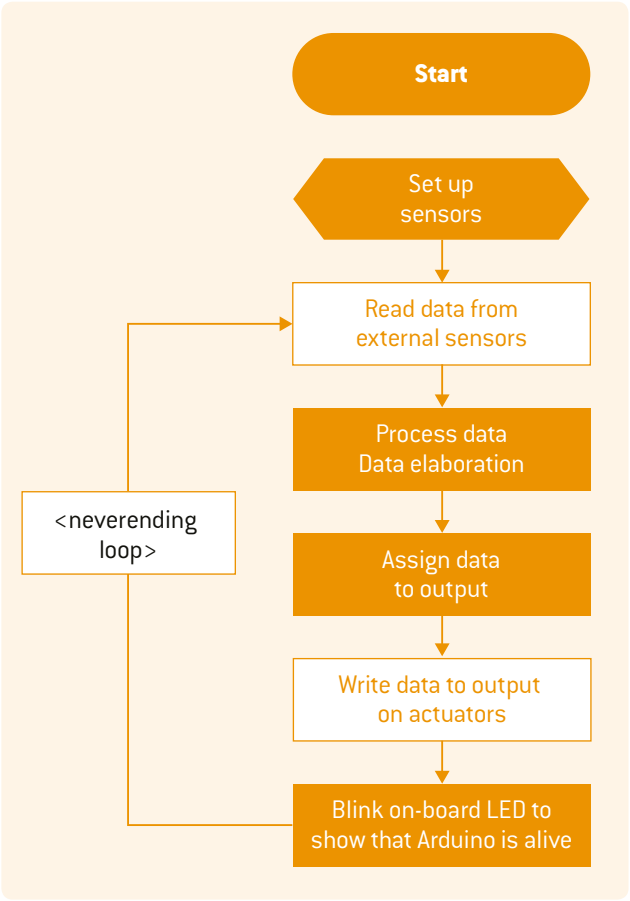
The code is normally based on an infinite loop (the machine is usually 'alive' as long as it is powered, and it must work all the time), where the three actions of measuring, processing and acting are performed in this order.

The students will learn that they can write any code with

1. sequential instructions
2. loops (while … do; repeat … until)
3. branches (if…then…else)

as stated in the Böhm-Jacopini theorem (see 'Additional information'[4]).

The second goal of the unit is to introduce the students to microcontrollers. They will learn to set up input sensors and output actuators, using digital or analogue ports, and how to write a simple program that reads input, processes data and writes output. They can choose to output either sound or light signals. These can be interpreted as 'alarms' which issue a warning based on the input read by the sensors.



3: Flow chart

The students will use the Arduino[1] integrated development environment (Arduino IDE[3]) to program in C++, with ready-to-use function libraries that make the coding process easier and faster.

A breadboard (see 'Additional information'[4]) must be available to allow the students to prototype and easily connect sensor pins to the Arduino I/Os, 5V power supply and GND pin.

The structure of any program, in metalanguage, is shown in 3.

Please note that the meta instruction *'While (TRUE)'* is a trick to tell any processor to repeat the included instructions indefinitely (as long as the microcontroller is powered).

The third main goal is to learn how to convert an acquired physical quantity into another quantity (e.g. light intensity into sound), ready for transmission to the external environment, by following these steps:

1. The physical signal (light, sound, force, energy …) is acquired by the sensor and converted into an electrical signal.
2. The electrical signal is transformed into a number available to the processor.
3. The number is processed and transformed by the processor into another number, and then used to do an action with an actuator transducer.
4. The actuators convert the number into electrical signals that are ready to be output.
5. The electrical signal is finally transformed into a physical signal (e.g. sound, light).

In 1 and 5, the signals must be converted from one form into another. In these phases, the linearity of the transformation, or the 'close to linearity' dependence, is extremely important (see 'Additional information'[4]).

See 'Additional information'[4] for a detailed explanation of the last line of the metacoding ('Blink on-board LED').

The input signal is usually called a 'stimulus' and comes from the environment where the sensors are placed to get data. The processor and the code are designed to 'react' to the stimulus with mathematical/logical operations (performed by the code instructions, processed by the microcontroller) and to output this 'response' to the environment. In our activities, the environment is the space surrounding the Arduino, which is able to 'see', 'hear' and 'feel forces' thanks to the sensors.

Working with the TI-Innovator Hub allows the students to focus more on the coding part of their work, as the microcontrollers and sensors are already set up and ready to use.

<What the students/teachers do>

We recommend that you begin by brainstorming to collect all your students' naïve ideas about sensors and automatic machine control. Collecting these ideas, gaining practical experience with the sensors and automatic machine control and then comparing the results with their previous (maybe incorrect) ideas is a good way to help the students to truly understand the process.

You could prepare a form with questions like:

- ↳ Do you know how a thermostat controls the temperature in a room?
- ↳ What is a sound warning-based parking system for? How does the driver react when the sound is emitted by the system?
- ↳ Do you have an induction hob in your kitchen? What does an illuminated LED mean?

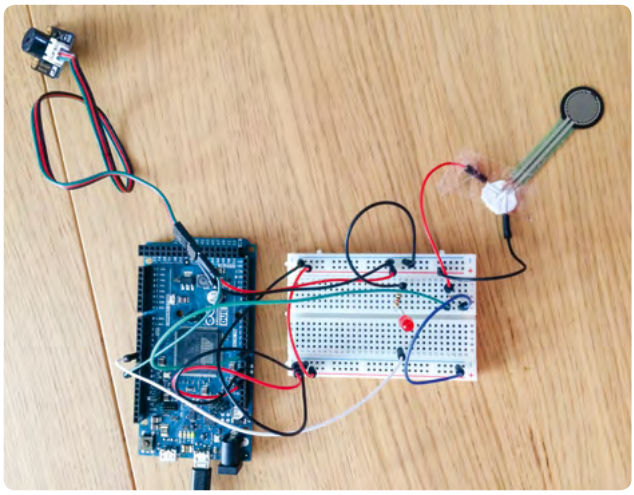
See 'PEC and list of questions'[4] for an example of a complete question list and references to the PEC (Prevision, Experience, and Correction) teaching methodology.

<Theoretical phase with Arduino[1]>

The teacher will introduce C++ programming[3] with the structure and basic instructions so that the students can write a simple loop using the instructions *analogRead, digitalRead, analogWrite, digitalWrite, if…then…else, loop, while*.

Hardware part

The teacher will present the microcontroller layout, showing the microprocessor, the analogue I/O pins (connections) and the digital input/output pins (connections).



4: Arduino, breadboard and sensors

The students will learn that any sensor/actuator usually has multiple connections:

↳ to the 5V or 3.3V Arduino output to get the power supply

↳ to the GND (ground) signal so that a current can flow

and

↳ to another digital or analogue input pin if it is used to read (get) external data

or

↳ to another digital or analogue output pin if it is used to take actions that generate an output, e.g. emit a sound or light, or do anything else that signals a situation (a 'write' operation)

Software part

The teacher will present a simple program that reads a sensor and writes an actuator, where a clear association between the physical pins and physical address on-board the microcontroller can easily be established by students.

An example of ready-to-use instructions is available online ('Program example 1'[4]).

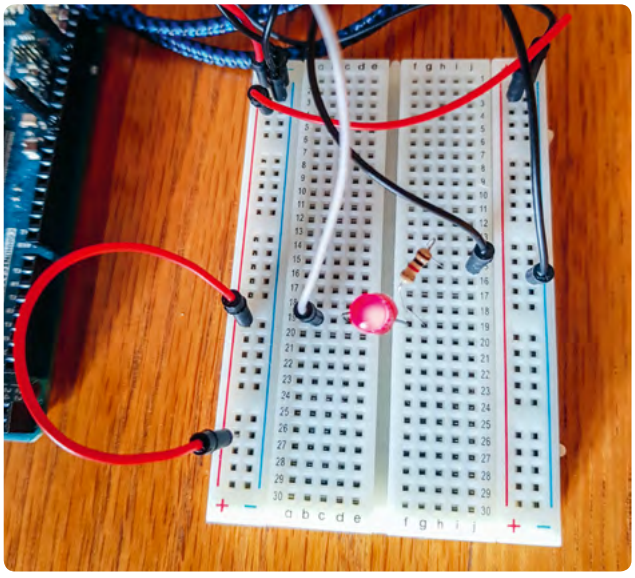
The students must keep in mind that the processor executes the code instructions one after the other and in the order in which they are written. Only the 'loop' instruction alters this principle as it tells the processor to continuously repeat the bracketed instructions as long as the microcontroller is powered.

<Practical phase with Arduino>

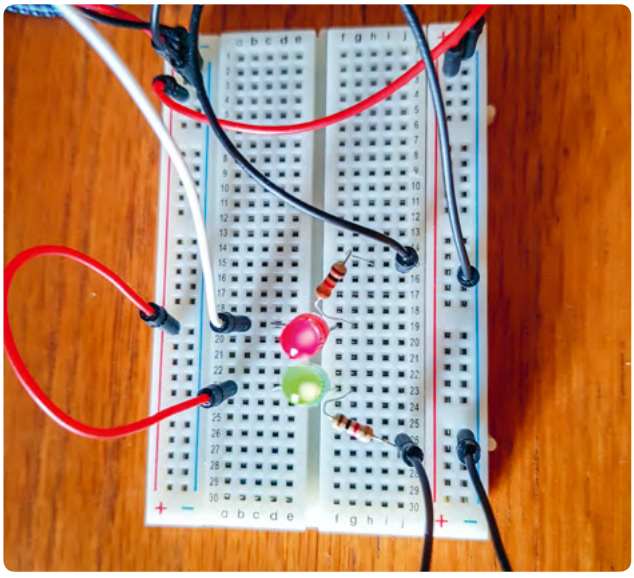
The students will get hands-on experience with the microcontroller, the breadboard and the sensors. The teacher should present the structure of the breadboard, showing all the available connections and how the students can take 5V, GND and I/O signals from the Arduino[1] to the breadboard. The students will be invited to copy the given coding example and to try, test and debug the code with the connected I/Os.

Hardware part

The students need the microcontroller, the sensors and short cables (10 cm) to allow easy connections between the sensor pins and the breadboard 'holes'. Sometimes you might need to solder additional cables to the sensors, but many sensors do not require this.



5: Breadboard detail



6: Breadboard with LEDs

The students will use the short cables to connect the 5V power supply and the GND signal to the breadboard, and the analogue/digital pins of the Arduino[1] to some 'holes' on the breadboard. This will allow the sensors to be easily lodged on the breadboard, and receive the required electrical signals (see 6).

Software part

After checking that the microcontroller-to-breadboard connections have been properly settled, with the exact correspondence between the pin logical number on the microcontroller and the sensor pin on the breadboard, the students will try to write the simple code provided ('Program example 1'[4]).

Algorithms with Arduino

We have prepared several signal conversions from one physical form to another.

Conversion of an analogue light signal into a digital light signal in an LED (modulated with a PWM feature) and a sound: the emitted sound frequency increases with the intensity of the light. Practical application: alarm clock, assistance for blind people. (See 7)

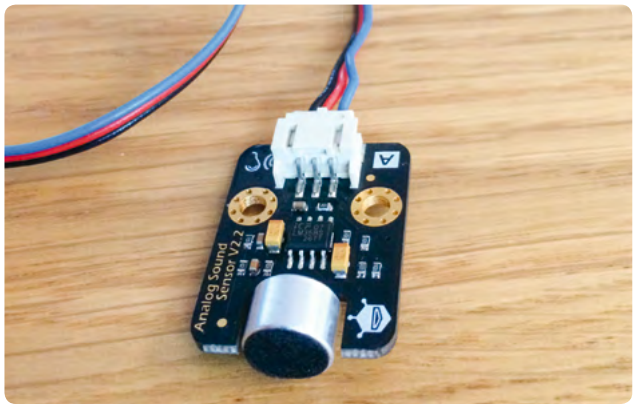
7: A light sensor

Conversion of a force detected through a force sensor into a digital light signal in an LED (modulated with a PWM feature) and a sound: the light intensity increases with the intensity of the force. Practical application: weight load alarm. (See 8)

8: A digital buzzer

Conversion of an external noise signal into a digital light signal in an LED. The higher the sound, the higher the light frequency will be. Practical application: noise pollution control.

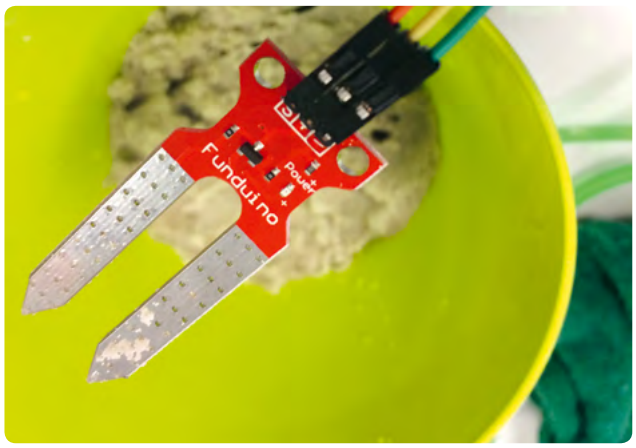
Conversion of a distance measured with a distance sensor into a sound. Technical application: car parking sensors. (See 9)



9: A distance sensor

Conversion of a temperature signal into a sound and a light signal. Technical application: oven temperature control.

Conversion of soil moisture concentration into a light signal. Technical application: plant irrigation and watering alarms and control. (See 10)



10: A soil moisture sensor

See 'Program example 2'[4] for the code used for these applications on the Arduino board.

All these algorithms serve as signal control and warning systems which monitor a selected environment on a physical quantity and issue a warning based on the input (stimulus) read.

<Theoretical phase with the TI-Innovator Hub>

Hardware part

The teacher can show the students how easy it is to connect the sensors.

Software part

The basic programming can be done on the calculator alone; the students only need to master the TI-Basic instructions provided above. Then the hub can be connected and the students will learn how to communicate with it, i.e. to use the instructions 'read' and 'get' to acquire data and 'set' to control the outputs.

<Practical phase with the TI-Innovator Hub>

Hardware and software part

The students will start with basic examples to familiarise themselves with the hub before working on more open problems. The students will learn how to control the different outputs with small exercises, e.g. controlling the colour of the LED, making it blink, controlling the duration of the blinking and producing sounds of a given frequency. The infinite loop will again be the basic structure to continue the operations indefinitely.

Algorithms with the TI-Innovator Hub

The students will solve two open problems: computing an automatic switch which turns on the light only if the ambient light intensity is lower than a certain threshold, and computing an alarm clock, which emits a sound of increasing frequency as the ambient light increases. Further developments are possible, but extra sensors will need to be bought and plugged into the hub.

<Buying the sensors>

Information on where and how to buy the sensors is available online.[4]

<Conclusion>

At the end of these activities, we noticed that our students' understanding of coding, the general program structure as well as logic and algorithms had improved significantly.

<Cooperation activity>

A wonderful cooperation activity could be to promote self-entrepreneurship. The students could try to invent an original human machine interface (HMI) which is technically useful. This HMI should read a stimulus from an environment (atmosphere, home, human body, etc.) and react by issuing another signal that emits a warning, performs an action or signals a situation. Partner schools abroad could carry out a market survey to gauge the market demand and value that the device may have in their countries. Any school taking part in this exchange could invent a device and make market enquiries for the product assembled by the other schools. At the end of the project, the most popular tool could be produced on a small scale by a partner company, and sold. Self-entrepreneurship is highly regarded all around the world, as it offers a great opportunity to teach science, technology and finance-related subjects together.

<References>

[1] www.arduino.cc
[2] www.arduino.cc/en/Guide/ArduinoDue
[3] www.arduino.cc/en/Main/Software
[4] All additional materials are available at www.science-on-stage.de/coding-materials.

CoALA-Code a Little Animal

<Author> Mirek Hančl
<Author> Julia Winckler

<Info>

<Keywords> simulation, IPO model (input-processing-output), measurement, computational thinking, making

<Disciplines> science, biology, computer science

<Age level of the students> 9–13

<Hardware> Calliope mini[1] or BBC micro:bit[2]
<Workshop A> crocodile clips, red craft plastic, USB cable and battery for the Calliope mini, self-adhesive copper tape (5 mm), cardboard, glue, scissors, small water glass, poster with animal pictures
<Workshop B> crocodile clips, USB cable and battery for the Calliope mini, Grove moisture sensor, Grove I2C touch sensor, Grove NFC, Grove I2C hub[3], cardboard, red craft plastic, small water glass, poster with animal pictures

<Language> MakeCode[4]

<Programming level> easy

<Summary>

You would be hard-pressed to find a child who does not want to own a pet. To find out which one is the best, the students will construct a simulator that is controlled by a single-board computer and uses external sensors to imitate the needs of a pet.

<Conceptual introduction>

The subject of 'pets' is not only part of the curriculum in primary schools but also in secondary schools in biology, where students learn how dogs were bred from wolves, the basic needs of a pet and the requirements that owners need to meet. Typically, students analyse texts in their schoolbook or videos on the Internet because schools cannot easily provide pets for this purpose. Therefore, an electronic simulator to depict the basic needs of a pet (food, drink, exercise, petting and correct body temperature) would be both illustrative and instructive.

The CoALA project does not utilise ready-to-use devices from commercial teaching material manufacturers that only allow specific, limited programs provided by the manufacturer. Nor is a simple toy used, such as the Tamagotchi, which was a worldwide success in the 1990s. Instead, the students plan, construct and program their own simulator in the form of their favourite pet, including an image of the animal, with the aid of a single-board computer (in our case, a Calliope mini[1] or a BBC micro:bit [2]) and craft supplies such as cardboard, copper tape, and external sensors. The students program an algorithm to record and evaluate the basic needs of the chosen animal. Depending on the algorithm, the animal simulator shows different smileys (to show how the animal feels) or plays fitting self-composed melodies.

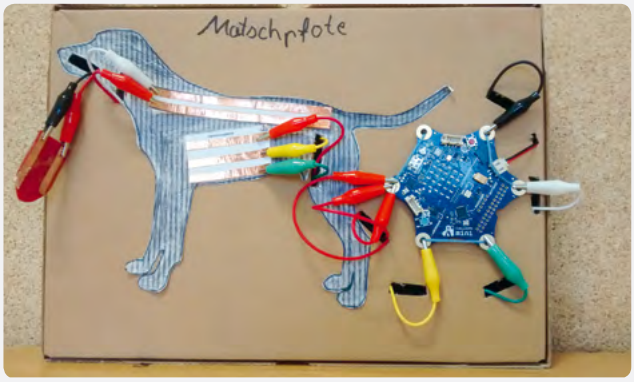
The concept of the project 'CoALA—Code a Little Animal' is that of a workshop. The OER (Open Educational Resources) teaching materials consist of three parts. The first part introduces the students to the basics of algorithms and handling the Calliope mini single-board computer[1]. In part two, they explore the basic needs of a pet and how to assess them. In part three of the workshop, the students build their favourite pet with cardboard, install the single-board computer and the appropriate sensors and create suitable algorithms with the aid of a graphical programming language.

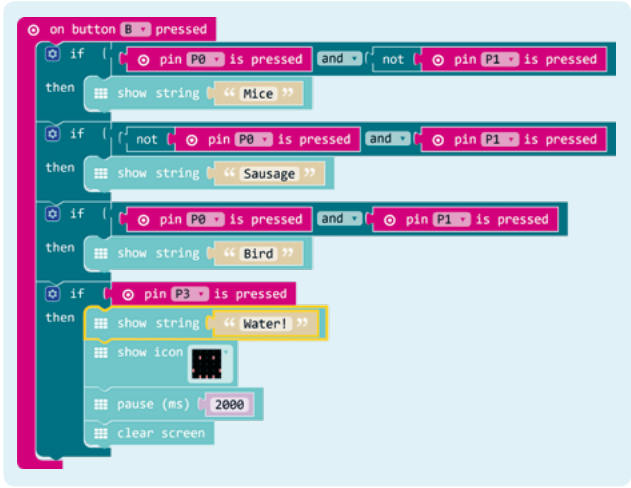
To meet the requirements of the science curriculum in primary schools and the biology curriculum in secondary schools, we offer the workshop materials in two versions. For primary schools (workshop A), the rates of eating, drinking and petting are measured and recorded by using conductive, adhesive copper tape. For secondary schools (workshop B), the students use external sensors to measure moisture (drinking), for multi-touch interactions (petting) and for the wireless read-out of Near Field Communication chips (eating). In both versions, built-in sensors measure movement and temperature.

All the workshop materials plus coding examples for the programming environment MakeCode[4] are available for download online.[5]

<What the students/teachers do>

To build the pet simulator, the students look for an image of their favourite pet or take a picture themselves. The printed image is glued onto the cardboard and equipped with adhesive copper tape (workshop A) or external sensors (workshop B) in the appropriate places. The adhesive copper tape or the external sensors is/are wired to the connections of the single-board computer and a suitable program is written on the computer to make it 'intelligent'. In the following example, the basic need 'food' is used to explain how the two versions of the workshop differ and how the programming environment is used.





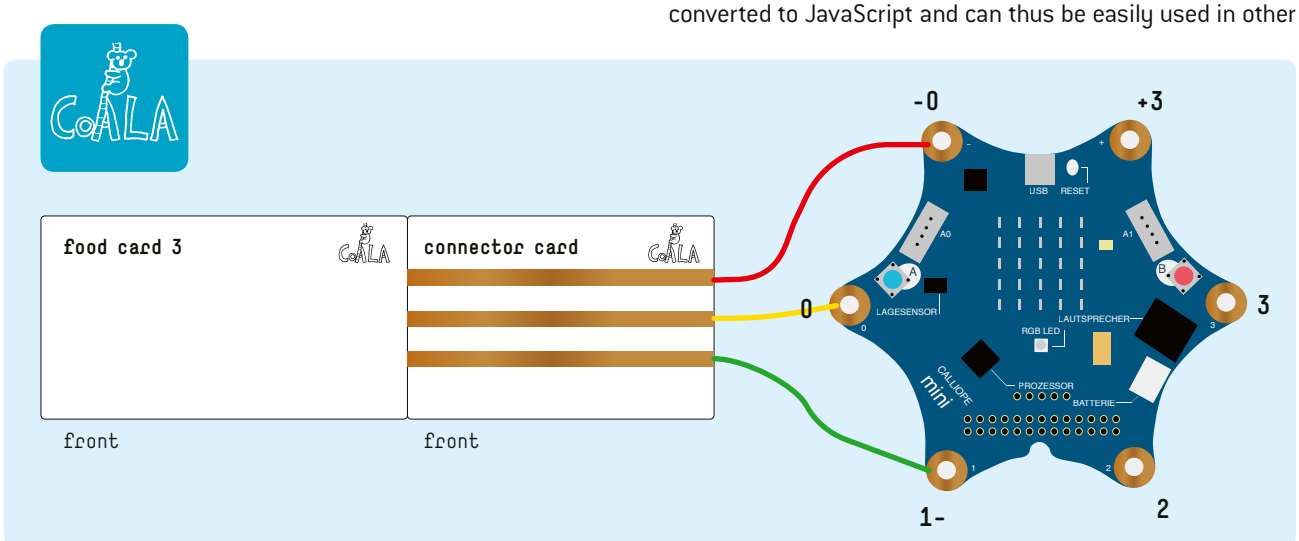
When the simulated pet feeds, taste sensors obviously cannot be used. Instead, the appropriate sensor 'reads' the offered food and the algorithm is controlled by a suitable conditional branch so that the output corresponds to the expected behaviour of the pet. Therefore, the display of the cat simulator shows a smiley (face) when it is fed a mouse and a sad face when it receives a bone. These branches are the same for both versions of the workshop.

However, the food sensors are completely different. In workshop A, pictures of different kinds of food are fastened to cardboard cards. On the other side, copper tape is glued to those cards so that the 'tongue' of the simulator 'reads' a binary coded number when the card is held to it. As the connections of the 'reader', i.e. 'tongue', are connected to the different pins of the single-board computer, the algorithm can test directly whether the pins are short-circuited or not: the food cards short-circuit different combinations of pins.

In workshop B, an external sensor with an NFC chip and attached radio antenna is used to wirelessly read out strings from an NFC tag. This tag can either be on an adhesive label or in a chip card. Unlike in workshop A, no binary coded number is read out, but instead the name of a food like 'fish' or 'bone'. This increases the implementation options and the complexity significantly. In the algorithm, the conditional branch is controlled by comparing the read-out value with the strings provided. In the CoALA project, the NFC tags are written on via a smartphone app; the read-out of the NFC tag is didactically reduced to a single block of code in MakeCode[4] and then loaded as an extension of the programming environment.

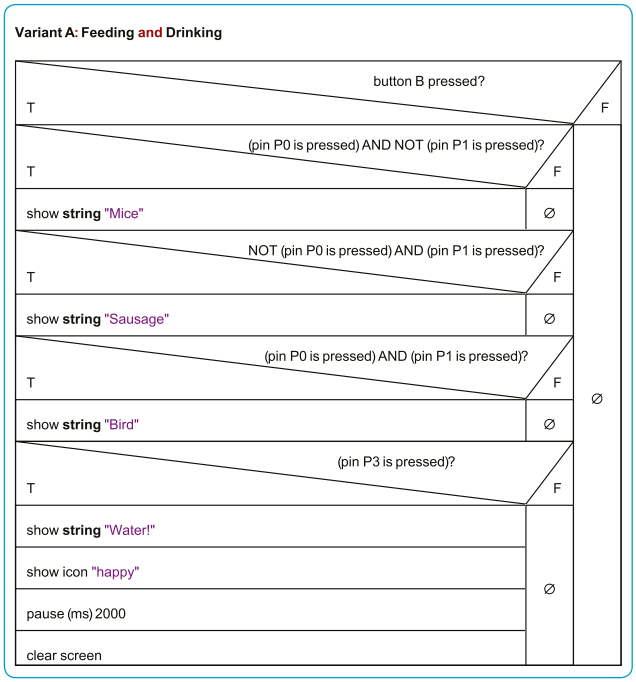
<Algorithm to use in other languages>

The available code examples[5] can be uploaded to the online MakeCode programming editor[4] and then used directly. By switching from block view to text view, the source code is converted to JavaScript and can thus be easily used in other



programming languages for the Calliope mini[1] or the BBC micro:bit[2]. The programming extensions for MakeCode used in the workshop materials to control the multitouch and NFC sensors also work for the BBC micro:bit.

Lastly, the programming examples are provided on the website as structure charts so the algorithms can be easily understood and ported to other platforms and programming environments such as Arduino.



<Conclusion>

The CoALA project provides students with the opportunity to familiarise themselves with the fundamental concepts of algorithmics—statements, sequences, conditional branching, loops and variables. They do not learn them by simple memorisation and reproduction but rather by working on an exciting educational project with real-life applicability. They use simple materials to construct a personal pet simulator, which they bring to life with the help of coding and their own imagination. The workshop materials provided teach computer science skills in a didactically reduced form and at the same time offer different levels of learning, thus meeting the needs of heterogeneous learning groups or older students. Both versions of the workshop can be easily mixed.

The workshop materials have been successfully tested with a Calliope mini[1] and the single-board computer BBC micro:bit[2]. A low cost extension board is needed to use the external grove sensors for moisture measurements, for NFC or for multitouch in workshop B with the BBC micro:bit.[3]

<Cooperation activity>

The CoALA workshop can be used in various forms of cooperation. As the material is for primary schools as well as for secondary schools, an exchange could take place between different types of schools. Thus, the exchange would not only be rewarding for the students but also for the teachers of both types of schools. The materials of workshop A are designed to teach simple logical Yes/No differentiations, while the materials of workshop B are designed to teach more complex, combined conditions, variables and string operations.

The materials of the workshops can also be combined as needed to promote cooperation within heterogeneous learning groups. Students who need more support could, for example, use the simple sensors of workshop A, while stronger students could explain the sensors of workshop B to their classmates and practise their communication skills in the process.

During the CoALA project, transnational cooperation between two secondary schools took place in which two groups of students—one from Germany and one from Spain—discussed their experiences with pet simulators via video conference. In addition to suggestions for problem-solving, the vocabulary of names and basic needs of their pets were exchanged in English as well as in their respective native languages—coding in STEM education with a language course.

<References>

[1] https://calliope.cc/en
[2] www.microbit.co.uk/home
[3] If you use a BBC micro:bit, you also need a Grove Shield for micro:bit.
[4] https://makecode.calliope.cc/?lang=en or https://makecode.microbit.org/?lang=en
[5] All additional materials are available at www.science-on-stage.de/coding-materials.

Liquid Data

<Author> Eleftheria Karagiorgou
<Author> Sevasti Tsiliki

<Info>

<Keywords> physical computing, acidity, water, liquid, temperature, pH, data logging

<Disciplines> chemistry

<Age level of the students> 16

<Hardware> Arduino starter kit[1], data logging shield, temperature sensor, pH sensor, SD card

<Language> Arduino IDE – Wiring C[2]

<Programming level> medium

<Duration of the project> 7 teaching hours

<Summary>

This teaching unit is an interdisciplinary approach using physical computing and chemistry. The students take on the role of researchers and conduct an experiment to determine whether there is a relationship between the acidity and the temperature of water. This will require the use of Arduino and chemistry.

<Conceptual introduction>

This educational activity was created to demonstrate to the students how physical computing can be integrated into STEM education, and more specifically, into chemistry by using innovative teaching methods. It was carried out as an extra-curricular activity during our school's Robotics and STEM club, which meets for two hours every Sunday afternoon.

The students took on the role of researchers and were tasked with proving the significant role that temperature plays in pH measurements. As the temperature rises, molecular vibrations increase, which allows water to ionise and form more hydrogen ions. This causes the pH level to drop as a result.

<Teaching method>

Inquiry-based science education: we wanted to involve our students in an active learning project, based on questions which then generate further questions as the students progress through the research project. In this way, the students become researchers and learn by doing a practical activity.

<Prerequisites–background knowledge>

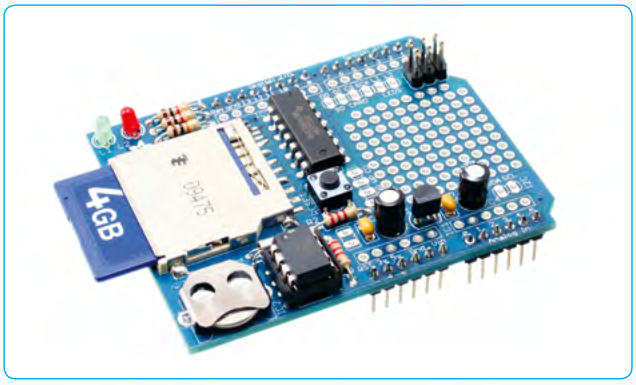
According to the Greek curriculum:

- basic knowledge of programming that was acquired during the 3rd year of junior high school and during the 1st year of senior high school
- basic knowledge of acidity and pH theory that was acquired during the 3rd year of junior high school

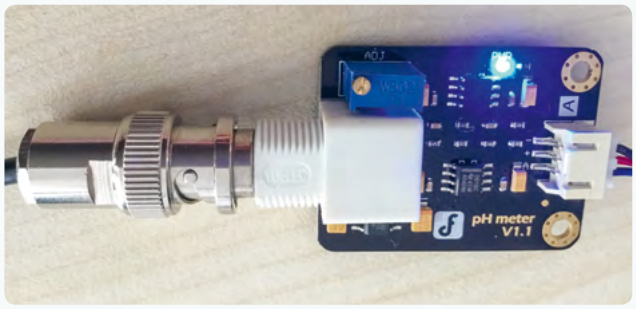
<Teaching materials/space used>

The Educational Robotics and STEM lab of the school contains the following:

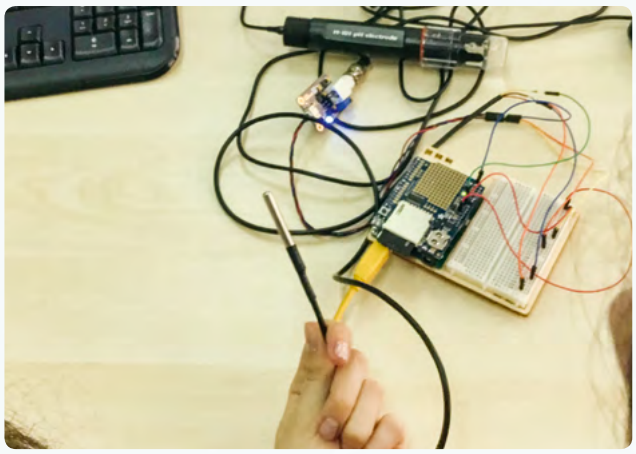
- Arduino[1] starter kit (includes an Arduino board, cables, LCD screen, etc.)
- Adafruit Data Logger Shield for Arduino (📷1)
- Analog pH meter Pro Kit for Arduino (📷2)
- Waterproof temperature sensor (📷3)
- SD card
- A computer with an SD port, such as a laptop, is necessary for the coding and the data logging parts of this teaching unit
- Demineralised water (purified water that has had most or all of its mineral and salt ions removed)
- Ice packs and a cooler to preserve the ice cubes



📷 1: An Adafruit Data Logger Shield for Arduino[3]



📷 2: Analog pH meter for Arduino



📷 3: Waterproof temperature sensor

<Research question>

Is there a relationship between the liquid's acidity and temperature?

<Problem-solving questions>

1. How can we connect the sensors with the Arduino board?
2. How will we do the data logging?

<What the students/teachers do>

<Preparatory phase: Introduction – theory - group work>

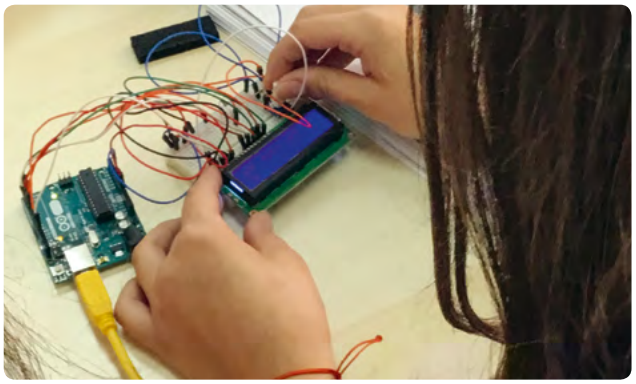
Duration: 1 hour

The students will be divided into groups and receive a short introduction to Arduino[1], the sensors used (pH sensor and temperature sensor) as well as how they work. They will also discuss the theory of acidity, the pH meter and the relationship between acidity and temperature. The students will then be asked to design an experiment to measure the level of fluctuation in the acidity of liquids as they change in temperature.

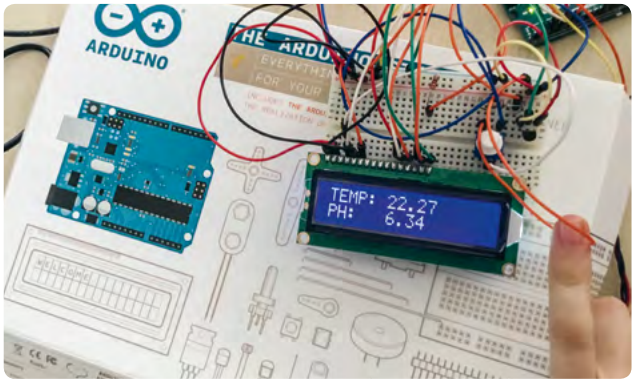
<Phase 1: Introduction to Arduino and how to code for it>

Duration: 1 hour

The students will familiarise themselves with the Arduino circuits and then learn the basics of how to code for Arduino. They will learn to connect the LCD screen to the Arduino and, through coding, display a message on it (📷4 & 5).



📷 4: Connecting the LCD screen



📷 5: Displaying sensor measurements

<Phase 2: Connecting the sensors>

Duration: 1 hour

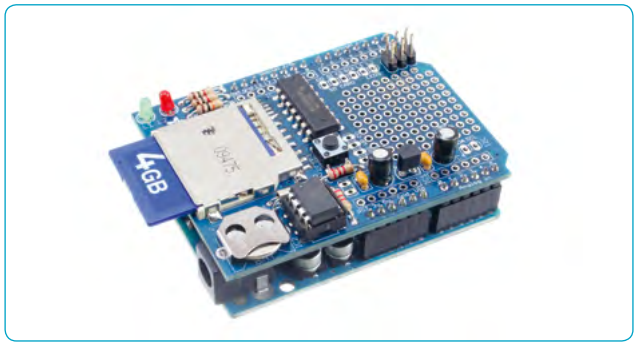
The students will learn how the pH sensor (📷6) and the temperature sensor (📷3) work by connecting them to the Arduino[1] and coding them to present the data on the LCD screen. This is a preparatory phase to understand the ins and outs of the sensors.

📷 6: A pH sensor

<Phase 3: The data logging shield>

Duration: 2 hours

The students will solder the data logging shield on the Arduino[1] board with the SD card to do the data logging (📷7). They will code the data logging shield, which has its own real-time clock (RTC). They will start the experiment by using demineralised water at 25 °C (neutral) and then they will measure the pH and the temperature by sinking the sensors into the liquid for 10 seconds.

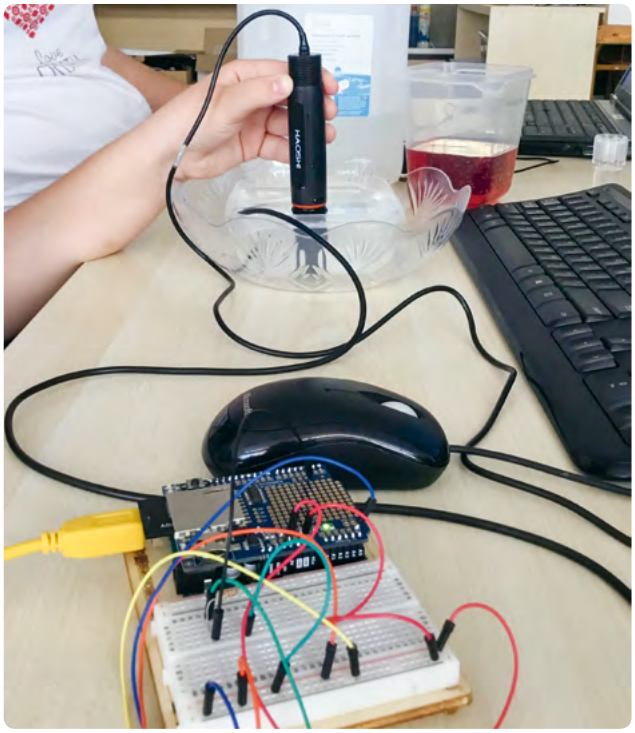


📷 7: Data logging shield soldered to the Arduino[4]

<Phase 4: The experiment>

Duration: 1 hour

The students will conduct tests with samples of the demineralised water, all of which have a different temperature. They will start with demineralised water (at room temperature) in a bowl, or beaker, cooled by surrounding ice cubes in a water bath (📷 8 & 9). Every 1 minute, they will sink the sensors into the liquid for 10 seconds. They will repeat the procedure at least 6 times to generate a large amount of data for the plotting phase of the unit. This way, the water bath will cool liquids gently and gradually.



8: Measurement of pH and temperature

9: A water bath with ice cubes

<Phase 5: Results>

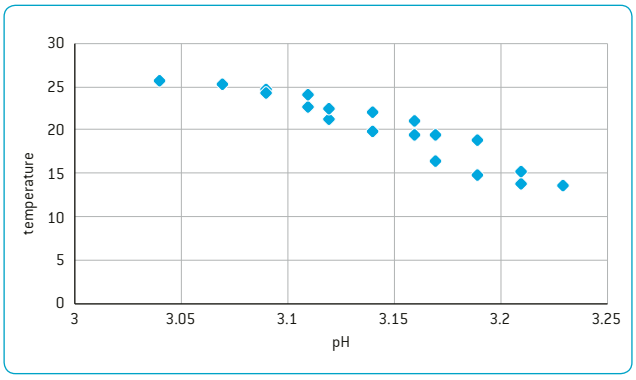
Duration: 1 hour

The students will unplug the SD card from the data logging shield, put it into the laptop to read the data file and do the plotting, which will illustrate the possible connection between temperature and acidity using a spreadsheet application (e.g. MS-Excel). The data will be saved on the SD card as a .csv file so it opens as a spreadsheet. The teacher will discuss the results of the data logging with the students and determine whether they managed to answer the problem-solving questions. The students will present their results to the class and discuss them with their peers.

<Conclusion>

By the end of the activity, the students are expected to understand the connections between the STEM subjects by implementing chemistry theory into a physical computing experiment. The students will also develop their inquiry-based thinking and working skills with the help of their teachers. Furthermore, the students will realise how the knowledge that they learn at school can be implemented practically in the real world as a result of doing these hands-on activities. Soft skills like co-operation, which are also developed while solving the questions and working on the projects, are essential for their future. Finally, this activity also offers the students an excellent opportunity to improve their performance in STEM-related fields and to better understand the importance of the cross-curricular nature of this project.

The experiment could be extended by using a wider variety of liquids. One example is vinegar in a water bath—either for a cold liquid, with ice cubes in the water bath, or for a hot one, with warm water (see 10).



10: Data of vinegar

The data logging shield must be precisely soldered to the Arduino board for this project. Some of your students may have difficulty doing this, so you should guide them through the process or even do the soldering yourself where necessary.

<References>

[1] www.arduino.cc

[2] www.arduino.cc/en/Main/Software

[3] Picture: oomlout (https://commons.wikimedia.org/wiki/File:ARSH-09-DL 03.jpg), „ARSH-09-DL 03“, CC BY-SA 2.0, https://creativecommons.org/licenses/by-sa/2.0/legalcode

[4] Picture: oomlout (https://commons.wikimedia.org/wiki/File:ARSH-09-DL_(5703636953).jpg), „ARSH-09-DL (5703636953)“, CC BY-SA 2.0, https://creativecommons.org/licenses/by-sa/2.0/legalcode

The Remote Captain

<Author> Immaculada Abad Nebot
<Author> Pere Compte Jové

<Info>

<Keywords> remote control, 2D and 3D design, modelling, electronic board soldering, chip programming, app programming, 3D printer

<Disciplines> technology, engineering

<Age level of the students> 14–16

<Hardware> Arduino[1], Bluetooth module, materials for building the model boat

<Language> Arduino, ArduinoBlocks[2], AppInventor[3]

<Programming level> medium

<Summary>

The students will design and build their own boat and navigate in a pool. Once they have completed this initial challenge, they will use an Arduino board to control the boat remotely with a tablet or smartphone.

<Conceptual introduction>

The students will design their own model boat and learn how to look at this task from an engineering perspective. The project will start with an analysis of the different kinds of boats using the Internet. After that, a small group of students will build a model that is stable on water.[4]

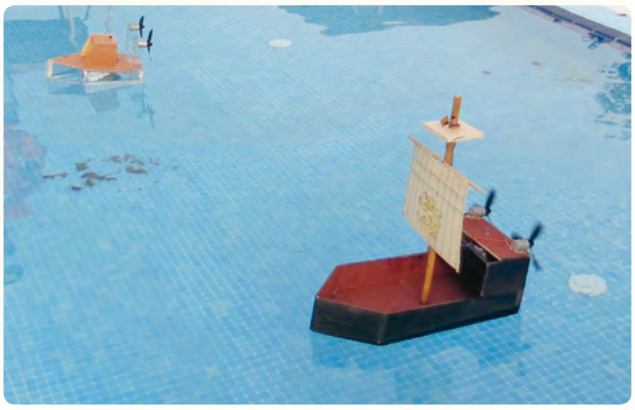
It will work with a reversing switch made by the students which lets the user control two motors (each one can turn forwards and backwards).

The students will incorporate a Bluetooth device so they can control the boat remotely with a smartphone. Using AppInventor[3], they will program a mobile app with different control systems, e.g. with buttons, with voice control or with an accelerometer (the boat will change direction according to the hand position of the person controlling it).

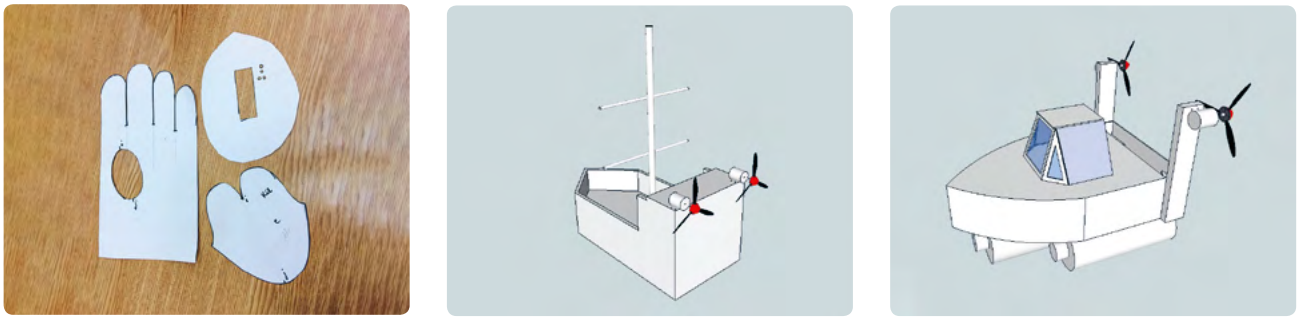
Finally, the students will have the opportunity to present their designs at a naval modelling exhibition (📷1) in front of experts, and to discuss any possible shortcomings in their models with them. This expert input will allow the students to improve their future models.

<What the students/teachers do>

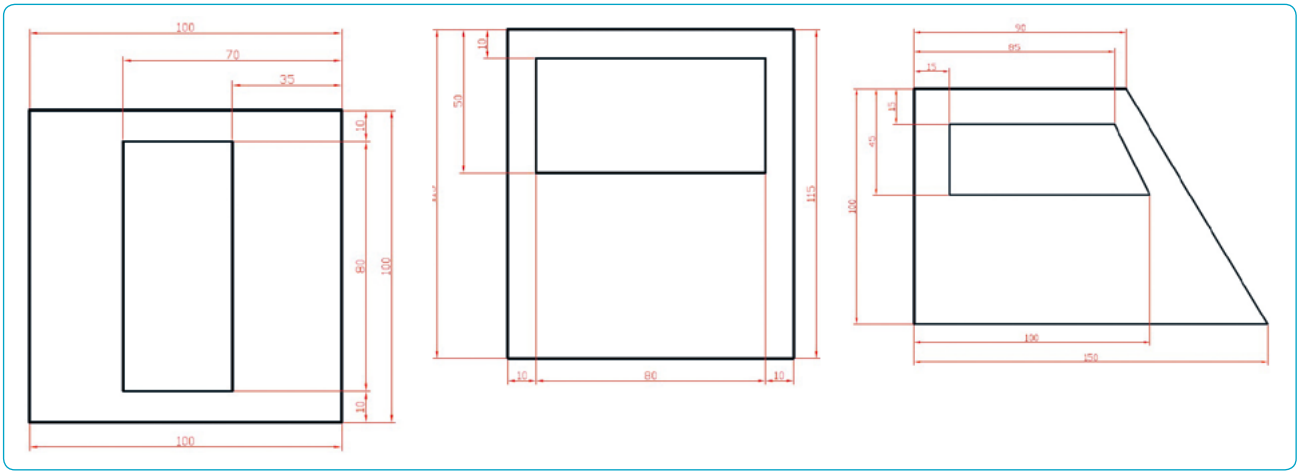
The students will create their design of choice after investigating different boat shapes on the Internet. They can draw the boat with a 3D design software such as sketchUp[5] or Tinkercad[6]. However, they need to keep in mind that the boats must be very stable on water to prevent them from overturning. (📷2)



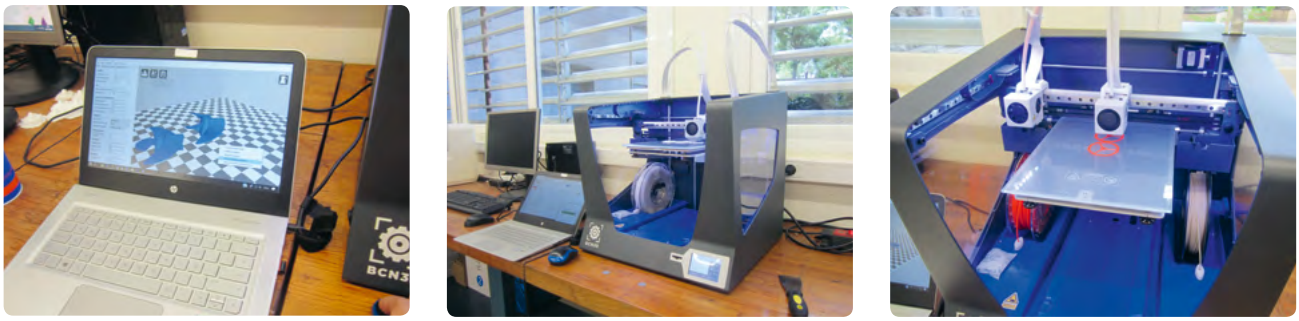
📷 1: Presentation of the model boats at a naval modelling exhibition in Spain



2: Various 3D boat models



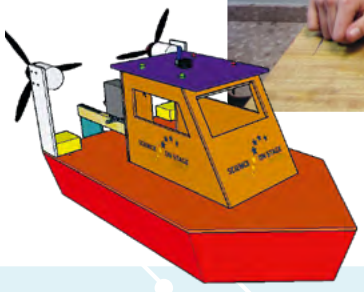
3: Door, front and side of the boat cabin



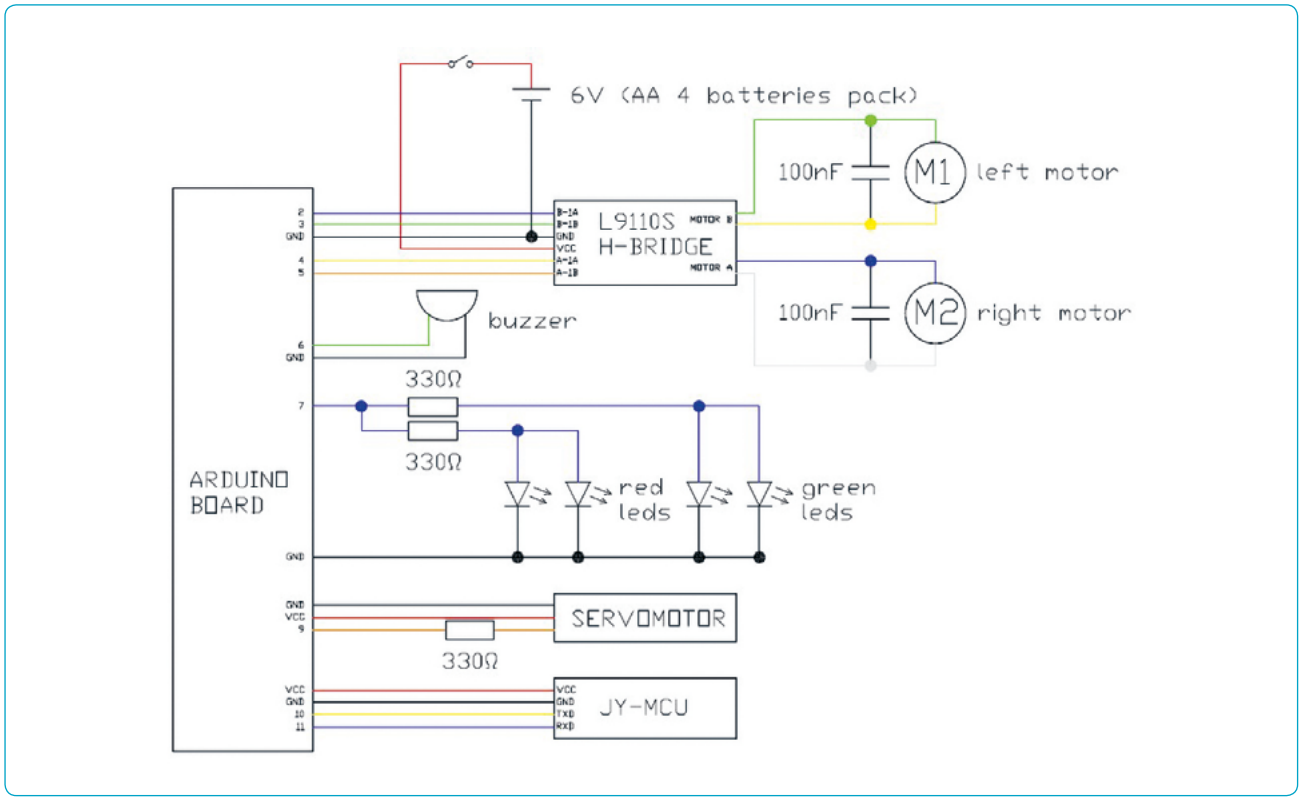
4a-c: Printing blades at the 3D printing centre Cesire Aulatec, Barcelona

We provide the plans for one model boat (3), including the required measurements, online. If you would like to change this design or print it, you can download various formats (.skp, .stl and .gcode).[4]

If your school does not have a 3D printer, it is often possible to cooperate with other institutions such as universities, maker spaces or similar. In our case, the students did their 3D printing at a 3D printing centre (4a–c). Step-by-step instructions for building the boat are available online.[4]



5, 6: Finished boat and 3D model

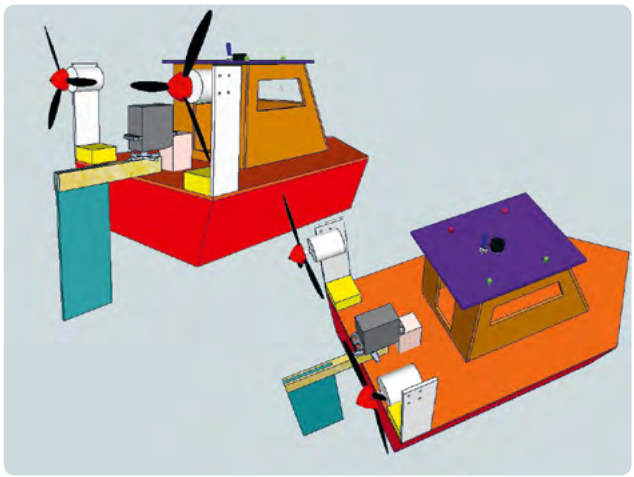


7: Electronic circuit diagram

<Electronic circuit>

To connect the accessories to the Arduino board[7], use the electronic circuit diagram in 7 and follow the respective instructions:

1. You can install a buzzer (Arduino pin 6) and lights (Arduino pin 7) on the roof. (8)
2. At the rear of the boat, you can connect a servomotor to control a rudder (Arduino pin 9). (8)
3. Connect the Bluetooth module in accordance with the schema TXD (Arduino pin 10) and RXD (Arduino pin 11).
4. Connect an L9110S motor driver controller board for Arduino with external batteries. (7)



8: The boat with accessories

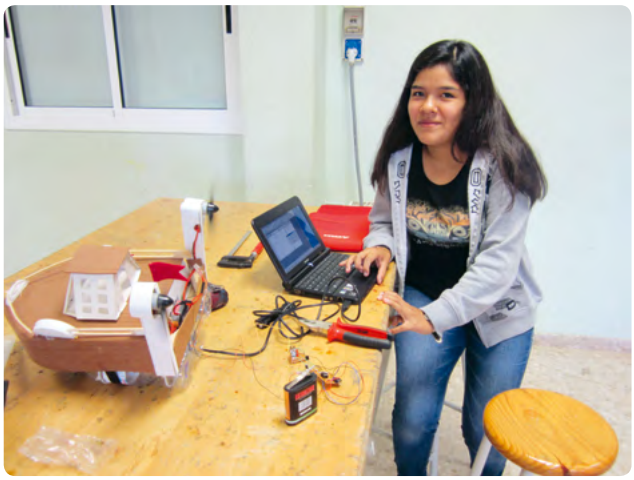
9: Construction of the boats

<How to control the motors and other features of the boat>

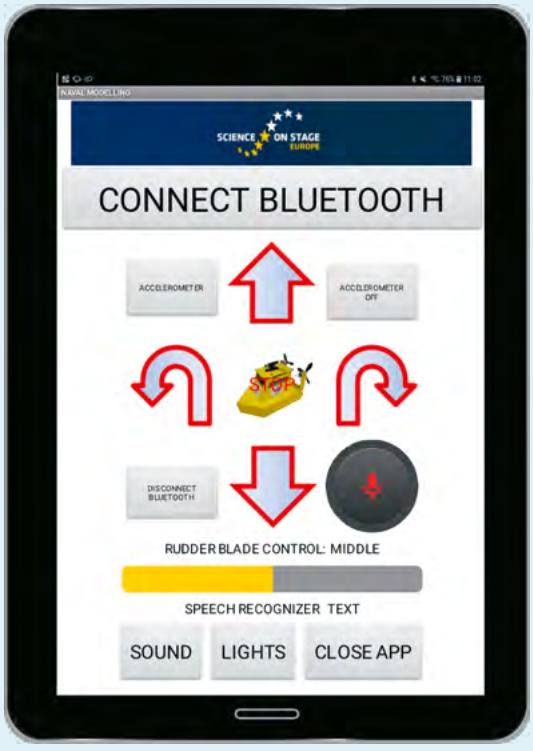
We recommend that you program the Arduino with Arduino IDE[1], ArduinoBlocks[2] or another similar program. The students can program the following tasks:

1. Connect and disconnect the lights on the roof.
2. Make a sound with the buzzer.
3. Control the servo position (40° right, 20° right, in the middle, 20° left and 40° left).
4. Both motors must turn and the boat moves forwards.
5. Both motors must turn and the boat moves backwards.

6. The boat must change its direction to the right (the left motor turns forwards and the right motor turns backwards).
7. The boat must change its direction to the left (the right motor turns forwards and the left motor turns backwards).
8. Control all the programs with Bluetooth.



10: Coding for the boats



11: The user interface of the app

<Program the app to control the boat with a smartphone using AppInventor[3]>

1. Program the app so it uses Bluetooth to connect with the boat.
2. Control the different elements of the boat with buttons.
3. Control the rudder blade with a scroll bar.
4. Control the boat using the accelerometer of the tablet or smartphone; tilting the smartphone forwards, backwards, right or left will cause the boat to move in the corresponding direction.
5. Use the option of speech recognition to control the boat with your voice.
6. Combine all these programs.

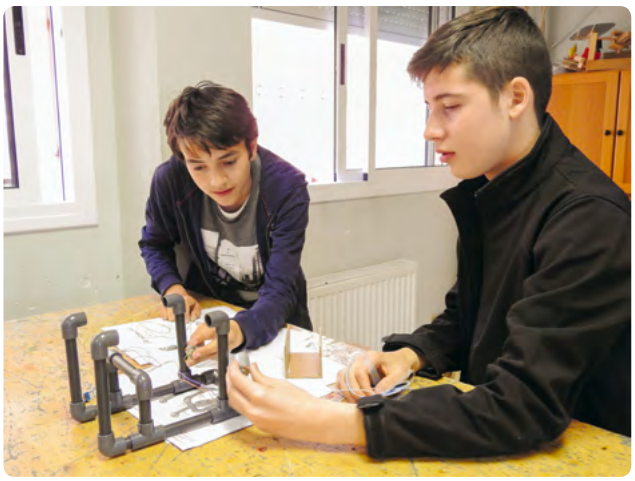
<Material list and equipment needed>

You can find a list containing all the required materials online[4]. The list includes details such as the quantity needed, price range and where the materials can be found.

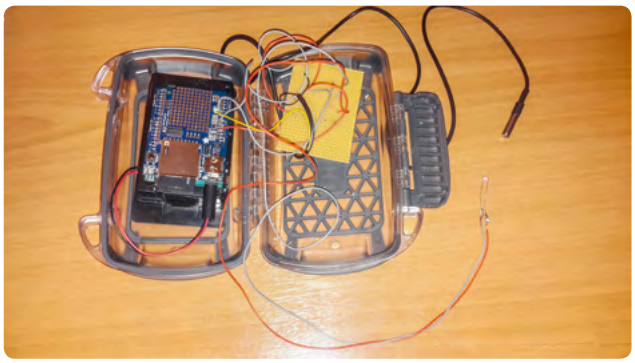
The material for one boat costs roughly €21. We spent approximately €15 on the electronics and €6 on the rest of the required materials.

<Cooperation activity>

While editing this material, we cooperated with Eleftheria Karagiorgou and Sevasti Tsiliki from the 7th Senior High School of Trikala, Greece, to implement an Arduino circuit in a hydrobot that allows us to navigate underwater. In this case, it is very important to protect the motors with wax and use protective as well as waterproof housing for the Arduino plate to prevent any water from entering the motors. We equipped our hydrobot 'Argolith'[8] with the microcontroller Arduino UNO to provide it with an electronic 'brain' and to record luminosity and temperature measurements underwater. The 'brain' also included a data logging shield, which offered us a real-time clock and a recording circuit for an SD card, where we saved the measured data.



12: Constructing the hydrobot



13: Arduino with waterproof casing

The online material contains an underwater video of the 'Argolith' hydrobot during testing in a river in Trikala, Greece.[4]

<References>

[1] www.arduino.cc

[2] www.arduinoblocks.com

[3] http://appinventor.mit.edu

[4] All the steps in this project and additional information: www.science-on-stage.de/coding-materials.

[5] www.sketchup.com

[6] www.tinkercad.com

[7] www.arduino.cc/en/Reference/Board

[8] A construction manual is available at http://seaperch.mit.edu/build.php.

How to Code

<Author> Bernard Schriek

This article deals with the use of coding in science. You may need special hardware and software to use the teaching units of this booklet. It is also helpful to have a basic knowledge of the fundamentals of programming. This chapter aims to provide you with a clear overview of the required information.

<Hardware>

Coding in science mainly deals with the measurement of physical and chemical quantities with sensors and the control of certain outputs. Typical sensors measure temperature, noise, light, distance, pH, button press, touch, etc. Typical outputs are an LED, a buzzer, a speaker, a motor, etc.

<Arduino>

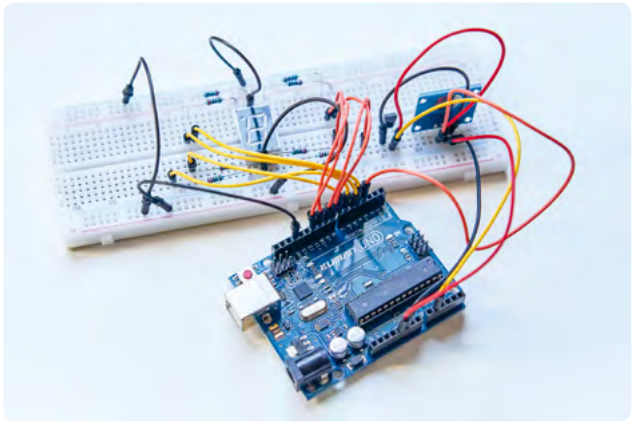
These sensors and outputs can be connected to or are already built into a microcontroller board or a single-board microcomputer. These can be more or less sophisticated. The smallest board is an Arduino[1]. It is available in different versions, but an Arduino UNO is widely used in schools (and also in this publication). The board contains an 8-bit microcontroller that runs quite slowly. However, the speed of the processor is not important for most applications as the board communicates solely via a reset button and an LED.

The board can be cabled to various other sensors and outputs. Programs for an Arduino are written on a computer and sent to an Arduino via a USB connection. When the program is loaded onto the Arduino, it can be started (and later be interrupted and restarted) by pressing the reset button. When the Arduino is connected via USB, it receives power through the USB connection; however, it needs a separate power source when it is disconnected from the computer.

One very practical way to expand an Arduino board is through the use of shields, which are circuit expansion boards that plug directly into the Arduino pin headers. For example, data logger shields that have an on-board real-time clock are often used. The data are written onto an SD card that is put into the SD card slot of the shield.

You can find well-documented programming examples on the Internet for almost every expansion (shields, sensors and outputs).

Windows, macOS or Linux computers can all be used to program the Arduino. The software can be downloaded online.[2] You will also find numerous in-depth examples on the Internet detailing how to program the Arduino. Programs for the Arduino are called 'sketches'.



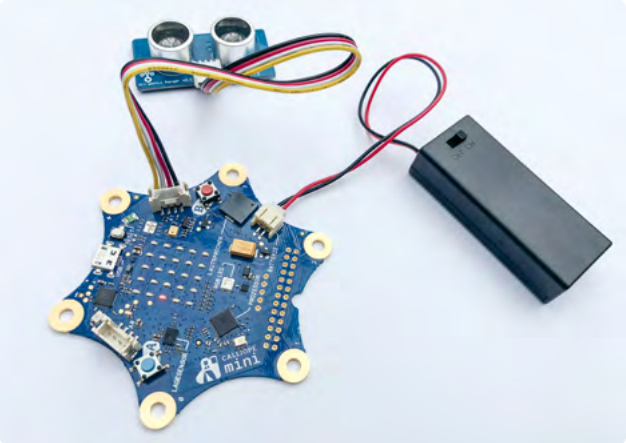
1: An Arduino

<Calliope mini and BBC micro:bit>

Another single board computer is the Calliope mini[3], which is compatible with the BBC micro:bit[4]. The difference between these two boards is that the Calliope mini has many sensors and actors already on-board, so for many projects you do not require external sensors and actors. The Calliope mini even has Bluetooth to communicate with other boards or smartphones. The processor is stronger and faster than the processor on the Arduino, and it also has considerably more on-board memory. Programs written on a computer can be transferred via USB onto the Calliope mini. The program starts automatically when it is transferred, but it can also be restarted by pressing the reset button. The Calliope mini has many sensors on board, but others can be attached by using standardised grove connectors[5].

It has a 5×5 LED matrix on-board that can be used for text scrolling. The Calliope mini is slightly more expensive than the Arduino, but the advantage of having many sensors and outputs on-board is worth the greater expense in everyday school life.

The Calliope mini uses JavaScript for programming, but you can also use block-oriented programming environments that are more suitable for younger children. They will be outlined later in this chapter.

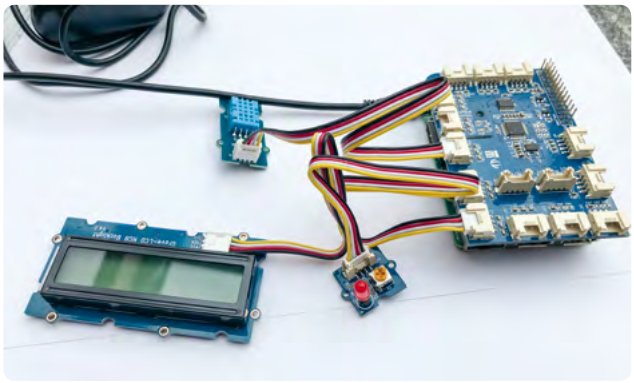


2: A Calliope mini with an ultrasonic distance sensor and a battery pack

<Raspberry Pi>

The Raspberry Pi[6] is a famous, fully functional single-board computer that runs both Linux and Windows as an operating system. It can be connected to a screen with an HDMI cable, and a mouse and keyboard can be connected via USB. At €50 to €60, its price is very reasonable, but still double that of a Calliope mini. You can use almost any programming language on the Raspberry Pi. Instead of a hard disk, it uses an SD card to save programs and data. It can be expanded with different expansion boards for different purposes. It has Wi-Fi on-board and can be linked into a local Wi-Fi network. Like all the other boards, it needs an external power supply either with batteries or a power adapter. While the Raspberry Pi is a full computer, it is not as fast as a normal desktop computer or laptop, and it only has an SD card as external memory. There are thousands of projects for it on the Internet. You should have some experience with computers to work with the Raspberry Pi.

3: A Raspberry Pi



4: A Raspberry Pi with an expansion board

<LEGO Mindstorms>

LEGO Mindstorms[7] is a much more expensive microcomputer system. Most students have some experience with LEGO, so they can easily build program-controlled tools and machines. LEGO has its own icon-based programming system called EV3 software, which is based on LabView[8] (a professional software for measuring and controlling). Here you move programming blocks to make a working program. Using and programming motors is very important when you program an EV3-Robot. Besides the original LEGO software, you can also use lejOS[9], a special implementation of the Java virtual machine. Due to Eclipse's existing LEGO plug-in, most people use Eclipse as the Java development tool on a computer. A large number of sensor and output bricks can be connected to the main processor brick, but like all LEGO materials, they are much more expensive than the sensors for the previously mentioned boards.

There are hundreds of other single-board computers, many of which are designed to run robots for every purpose. If you are looking for project ideas, we recommend that you visit

hackster.io[10]. However, please be aware that you need to register for free to get access to many project descriptions.

<Software>

In the 21st century programming and coding skills will become increasingly important in all areas of life. One goal of this booklet is to help teachers to encourage their students' interest in coding. Students are usually very interested in coding and they only need the right tools and programming languages to be successful and more motivated as a result. The choice of programming language depends on the age of the students. Younger students will need more visual help during the programming and debugging process. Therefore, the next paragraphs will outline how to implement several important coding concepts, using block and text programming.

<Variables>

Variables are used to hold values for later use. A good way to look at variables is to imagine them as boxes. A box holds a value, it has a certain form that depends on the type of value (integer, float, string, Boolean, i.e. true or false), and it has a label attached to it containing the name of the variable. We suggest that you use explanatory names for variables, such as *temperature* instead of the letter *t*, so other people can better understand your programs. In block-oriented languages (Scratch[11], Snap![12], MakeCode[13]), a variable looks like a label and you can watch the name and the value even while a program is running (📷5). This is very helpful for debugging. There are also blocks for setting or changing the value of a variable that are self-evident.

📷 5

<Assignments>

In some text-oriented programming languages, variables must be declared. The declaration includes information about the type of variable (integer, string, …). Other text-oriented languages wait for the first assignment to decide implicitly what type the variable is.

And the assignment itself has a hurdle: it is written like a mathematical equation. 'temperature = 23' is an assignment and means that the variable temperature receives the value 23 (📷6). If you want to compare the temperature with the value 25, you write 'temperature == 25', using two equal signs. This difference is the cause of many bugs in software development.

📷 6

<Program flow>

Most computer programs consist of statements that are consecutively processed, i.e. one after the other. Admittedly, many languages include parallel processing, where two or more programs run at the same time in threads—even Scratch[11] or Snap![12] offer this parallel processing.

But we will now look at a single process. The statements are executed from the first block or line to the last block or line. This is called a sequence. But things are not usually that simple: You want the program to carry out different commands based on decisions made at certain points in the program. This is called branching.

<Branching>

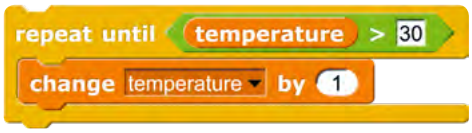
There are three types of branches: one-sided, two-sided and multi-sided, and they all need a condition to indicate what you want the program to do. This condition is usually a comparison between values. The result is a Boolean true/false. The one-sided branch just adds several additional statements to the program flow that are executed when the condition results to true. The two-sided branch adds two additional sets of instructions from which only one set is executed, depending on the result of the condition. The multi-sided branch takes a variable, and depending on the value of that variable, different sets of instructions are executed. Block-oriented languages provide a good visualisation of the condition and the instruction sets. Text-oriented languages use if- or if-else statements for the first two types of branches and case-statements for multi-sided branches.

7

8

<Loops>

Loops, which are used for repeatedly executed statement sets, are another important instrument that is used to control the flow of instructions. Loops can be distinguished by the condition that controls the loop. The condition—usually a comparison—can be at the beginning or the end of the loop. If it is at the beginning and false, the loop statements will never be executed. If the condition is at the end of the loop, the loop statements will be executed at least once. Counting loops (sometimes called for-loops) are also used when it is clear how many times the loop statements should be executed.



9

10

Statement sequences, loops and branches are normally nested. This means that a loop can be inside a branch, and a branch can be inside a loop. There are rules on how to document the program flow in a diagram. In this booklet, we use Nassi-Shneiderman diagrams[14] to explain the flow of instructions in our programs.

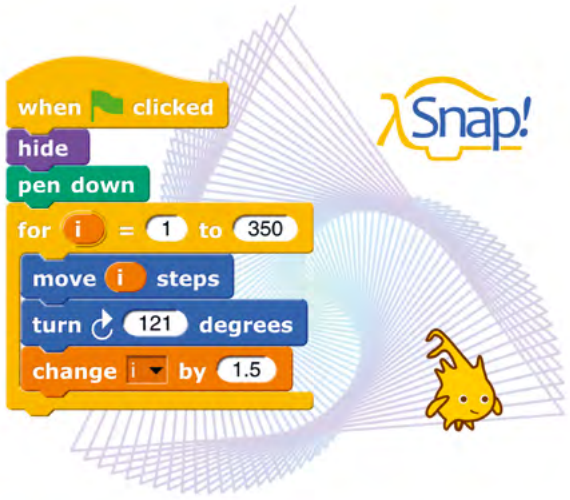
<References>

[1] www.arduino.cc
[2] www.arduino.cc/en/Main/Software
[3] https://calliope.cc/en
[4] www.microbit.co.uk/home
[5] http://wiki.seeedstudio.com/Grove_System/
[6] www.raspberrypi.org
[7] www.lego.com/en-us/mindstorms
[8] https://en.wikipedia.org/wiki/LabVIEW
[9] www.lejos.org
[10] www.hackster.io
[11] https://scratch.mit.edu
[12] https://snap.berkeley.edu
[13] https://makecode.calliope.cc/?lang=en
[14] https://en.wikipedia.org/wiki/Nassi-Shneiderman_diagram

<Computer Science Education with Snap!>

snap.berkeley.edu/run

Creativity as well as computational and media literacy are considered important skills in the ongoing digital revolution. *Snap!* is a tool that supports people of any age or background to get in touch with computer science (CS) and actively shape the current developments.



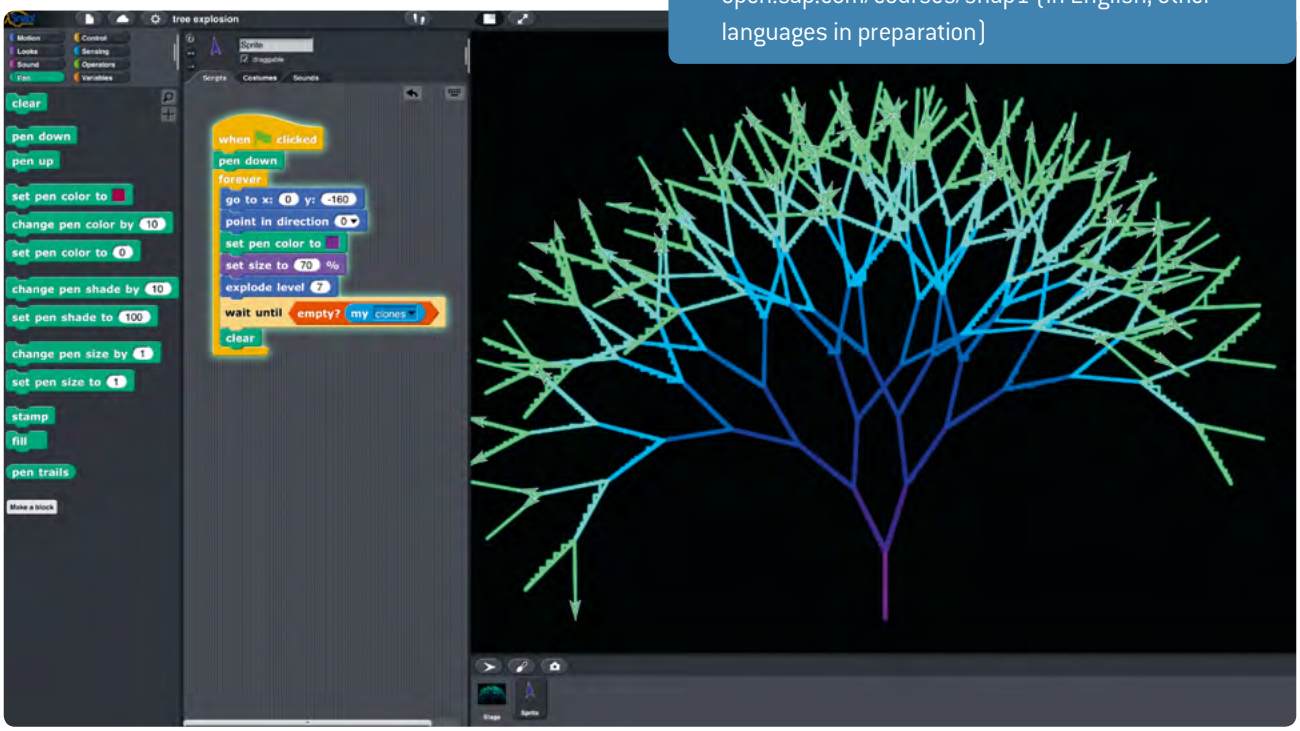
<What is Snap!?>

Snap!–Build your own Blocks is a visual, blocks-based programming language. It invites learners to bring their ideas to life while getting to know computer science in a playful and experimenting mode. Fascinating about *Snap!* is its keen aspiration to provide a low entry level while not reducing its expressiveness. It allows beginners and experienced programmers to immerse into advanced CS concepts like arbitrary data structures, higher order functions and even custom control structures in a visually appealing and understandable way.

Snap! is developed at SAP together with researchers of UC Berkeley. Today, it is available in over 40 languages and used for CS education in just as many countries. *Snap!* is open-source and runs in all modern web browsers.

<Did you know that Snap!…>

- is one of the top 100 programming languages in the TIOBE Index
- is used for artificial intelligence by researchers at the University of Oxford
- is used by the maker scene for 3D-printing, embroidery and robotics
- is easy to teach, for example with the free course on open.sap.com/courses/snap1 (in English, other languages in preparation)



<Participate in Meet and Code>

Get the support that you need to organise and promote coding!

'I had no idea that programming was so easy!' *Meet and Code 2018* was a real eye-opener for Philip. 'I really want to continue,' said Louisa after attending a hackathon. And that's exactly what the initiative wants to show; programming is fun and easy to learn.

In the second year of *Meet and Code,* more than 52,000 boys and girls participated in over 1,100 events in 22 countries. As always, the action took place during the EU Code Week in October.

© Peter Böhmer

Every audited event deemed worthy of funding received seed money of up to €500. All kinds of programming events can be supported and any non-profit organisation can apply with a project.

The initiative intends to continue the successes of last year in 2019: numerous events, projects and workshops will take place during the EU Code Week in 22 countries. The goal is to introduce children and young people between the ages of 8 and 24 to the world of technology and coding. The events are designed to show young people how much fun coding can be

© Dietrich Bechtel

and how it can help bring ideas to life. By exploring a broad range of technology and digital topics and creative coding, they will be encouraged to develop the digital skills they need in today's world.

The Munich based organization Haus des Stiftens gGmbH with its IT portal Stifter-helfen and the respective country partners of the TechSoup Europe network are behind the initiative. *Meet and Code* is made possible by SAP.

In 2019, prizes will again be offered for the most creative event ideas and most original implementations. The *Meet and Code Award* will be presented in at least three categories, including innovation and diversity.

All activities, information and registration at
www.meet-and-code.org

Follow us and join the discussion:
@stifter_helfen @TechSoupEurope
#meetandcode #codeEU #SAP4Good

© Dietrich Bechtel

© Peter Böhmer

<Further Material>

The authors have created additional resources and material for the teaching units. You can find them online as free download at **www.science-on-stage.de/coding-materials**

<Project Events within Coding in STEM Education>

<2 June 2016>
Brainstorming for topics within the final meeting of the project *Football in Science Teaching,* Brussels, Belgium

<20-22 October 2017>
First workshop in connection with the *European STEM League* Final in the German Football Museum, Dortmund, Germany

<13-15 April 2018>
Second workshop in Berlin, Germany

<1 March 2019>
Presentation of the publication in Vienna, Austria

<March 2019 - March 2020>
Code League Competition

<Follow-ups throughout 2019 and 2020>
Teacher training courses in various European countries

<Autumn 2020>
Code League Final

CODE LEAGUE <Coding in STEM>

SCIENCE ON STAGE EUROPE

Science on Stage - The European Network for Science Teachers

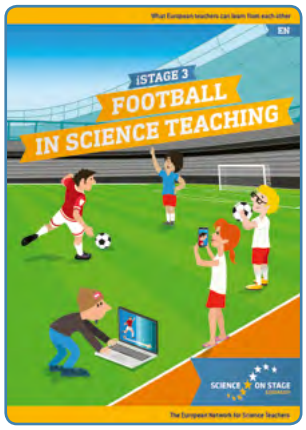
… is a network of and for science, technology, engineering and mathematics (STEM) teachers of all school levels.
… provides a European platform for the exchange of teaching ideas.
… highlights the importance of science and technology in schools and among the public.

The main supporter of Science on Stage is the Federation of German Employers' Association in the Metal and Electrical Engineering Industries (GESAMTMETALL) with its initiative think ING.

Join in – find your country on
www.science-on-stage.eu
www.facebook.com/scienceonstageeurope
www.twitter.com/ScienceOnStage

Subscribe for our newsletter
www.science-on-stage.eu/newsletter

<Further material>



Football in Science Teaching
↳ Teaching units about the various aspects of STEM in football
↳ Chapters: Biosphere, Body, Ball, Big Data

Smartphones in Science Teaching
↳ Guidelines and experiments for inquiry-based learning with smartphones

Lilu's House - Language Skills through Experiments
↳ Primary school students discover natural scientific phenomena in bathroom, living room and kitchen while they practice speaking, writing and reading.

Download free of charge at
www.science-on-stage.eu/teachingmaterials

A project by

SCIENCE ON STAGE GERMANY

Main supporter of Science on Stage Germany

think ING.
Die Initiative für Ingenieurnachwuchs

Proudly supported by

SAP®

www.science-on-stage.de