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>

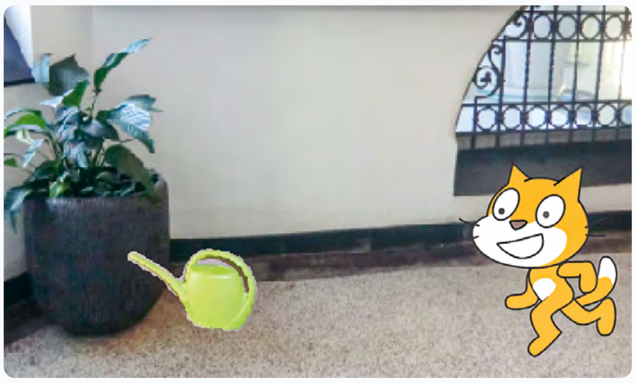
The potted plants in school buildings often die during the summer break because nobody takes care of them—this is why we need a Vacation Plant Life Saver. In this teaching unit, we will develop a virtual and a real-life lifesaver for school plants.

<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).

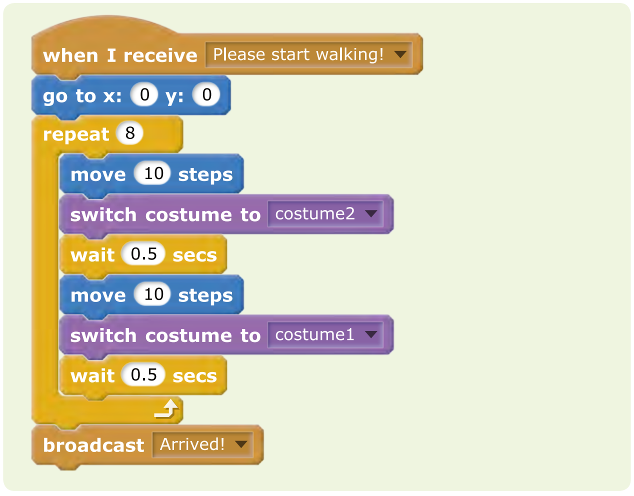
After analysing the problem ('How could a virtual Vacation Plant Life Save work?'), the students will be asked to think about the basic structure of such a program. They will then make notes in form of a recipe (algorithm) and test each other's ideas. Only after doing so will they agree on a common basic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

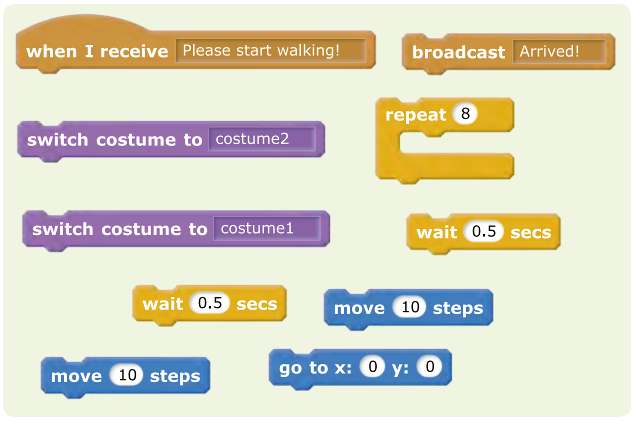
- ↳ object orientation (each figure has its own script, even the stage)
- ↳ structure (What does the structure of a condition or a loop look like in Scratch?)
- ↳ script/costume/sounds can be assigned to each individual figure

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.



```
when I receive [Please start walking! ▾]
go to x: (0) y: (0)
repeat (8)
    move (10) steps
    switch costume to [costume2 ▾]
    wait (0.5) secs
    move (10) steps
    switch costume to [costume1 ▾]
    wait (0.5) secs
end
broadcast [Arrived! ▾]
```

📷 2



```
when I receive [Please start walking!]      broadcast [Arrived!]

                                            repeat (8)
switch costume to [costume2]

switch costume to [costume1]                wait (0.5) secs

          wait (0.5) secs           move (10) steps

move (10) steps          go to x: (0) y: (0)
```

📷 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).

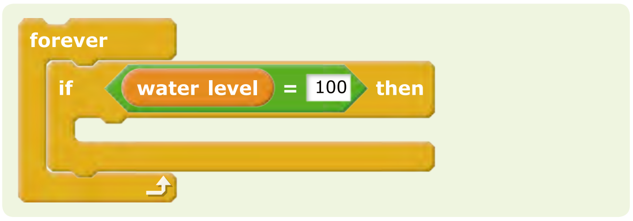
As an introduction to the second step of the teaching unit, the students will think about other factors that determine how often a potted plant must be watered. The 'room temperature' and the 'water level' in the plant container will certainly play a role here as changing variables (see Worksheet 2[5]).

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.



```
forever
    if <(water level) = [100]> then
    end
end
```

📷 4

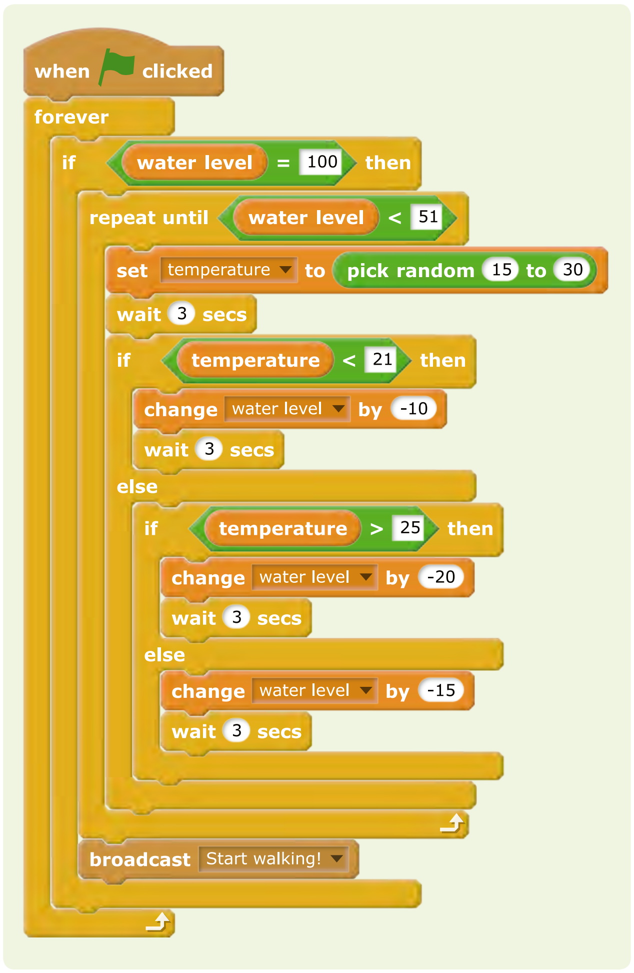
The ability to structure is fundamentally important in learning any programming language, but it will be taught in a very fun way; the students will be able to try out everything without any negative consequences.

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 reassembled 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).



```
when [flag] clicked
forever
    if <(water level) = 100> then
        repeat until <(water level) < 51>
            set temperature to (pick random 15 to 30)
            wait 3 secs
            if <(temperature) < 21> then
                change water level by -10
                wait 3 secs
            else
                if <(temperature) > 25> then
                    change water level by -20
                    wait 3 secs
                else
                    change water level by -15
                    wait 3 secs
        broadcast Start walking!
```

📷 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>

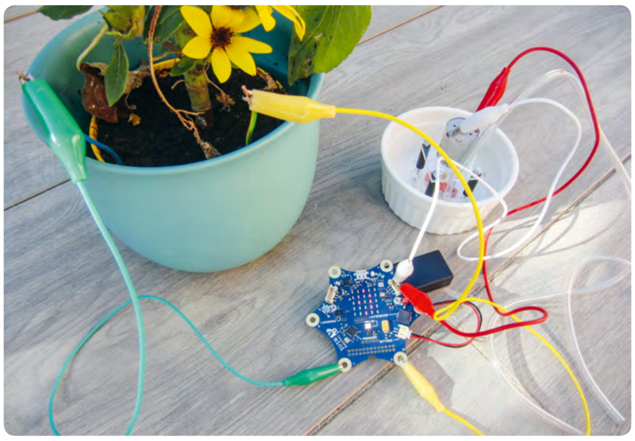
During the project, it quickly became clear that some students were not satisfied with the virtual solution to the watering problem. They asked about ways to water real plants with the help of their program. The virtual watering program is relatively easy to transfer to a real VPLS using a microcontroller, 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).



📷 6: VPLS controlled by a Calliope mini

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.

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/

<Imprint>

<Taken from>
Coding in STEM Education
www.science-on-stage.eu/coding

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

<Revision and Translation>
Translation-Probst AG

<Design>
WEBERSUPIRAN.berlin

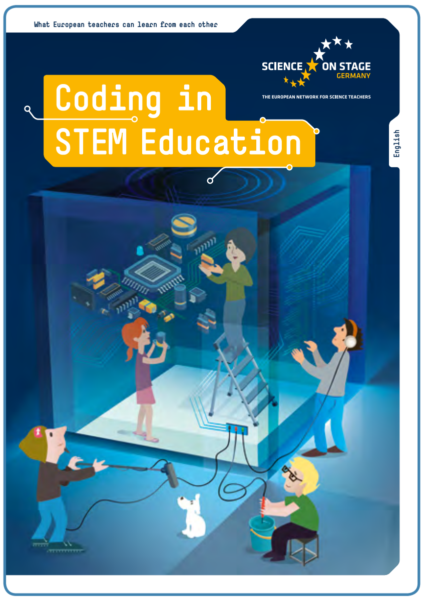
<Illustration>
Rupert Tacke, Tricom Kommunikation und Verlag GmbH

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

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

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

First edition published in 2019
© Science on Stage Deutschland e.V.


What European teachers can learn from each other
SCIENCE ON STAGE GERMANY
THE EUROPEAN NETWORK FOR SCIENCE TEACHERS
Coding in STEM Education
English

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

A project by

Main supporter of Science on Stage Germany

Die Initiative für Ingenieurnachwuchs

Proudly supported by