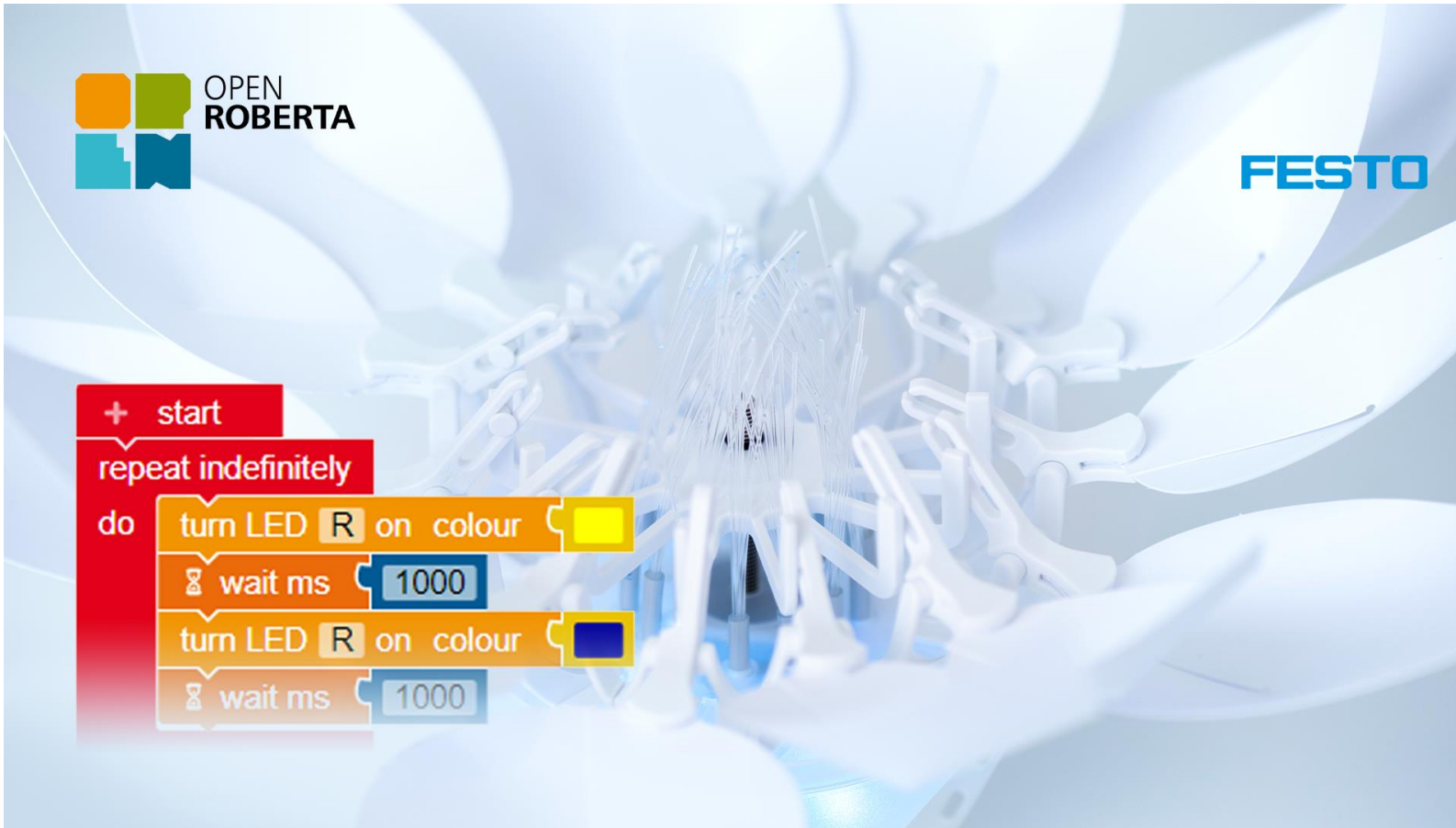


## Bionics4Education: Bionic Flower Courseware for Open Roberta



The image shows a screenshot of the Open Roberta programming environment. The background is a 3D rendering of a white bionic flower with a central sensor array. In the top left corner, there is the Open Roberta logo (four colored squares: orange, green, cyan, blue) and the text "OPEN ROBERTA". In the top right corner, the "FESTO" logo is displayed. The main area shows a Scratch-style script:

- start** (red block)
- repeat indefinitely** (red block)
- do** (red block) containing:
  - turn LED R on colour** (orange block) with a yellow color swatch.
  - wait ms** (blue block) with the value "1000".
  - turn LED R on colour** (orange block) with a blue color swatch.
  - wait ms** (blue block) with the value "1000".

# Bionic Flower: Introduction into Coding with Open Roberta

## Chapter structure

1. Introduction Bionic Flower & Biological Background Information
2. Introduction Hardware Bionic Flower
3. Introduction Programming
4. Introduction Open Roberta
5. DIY Projects



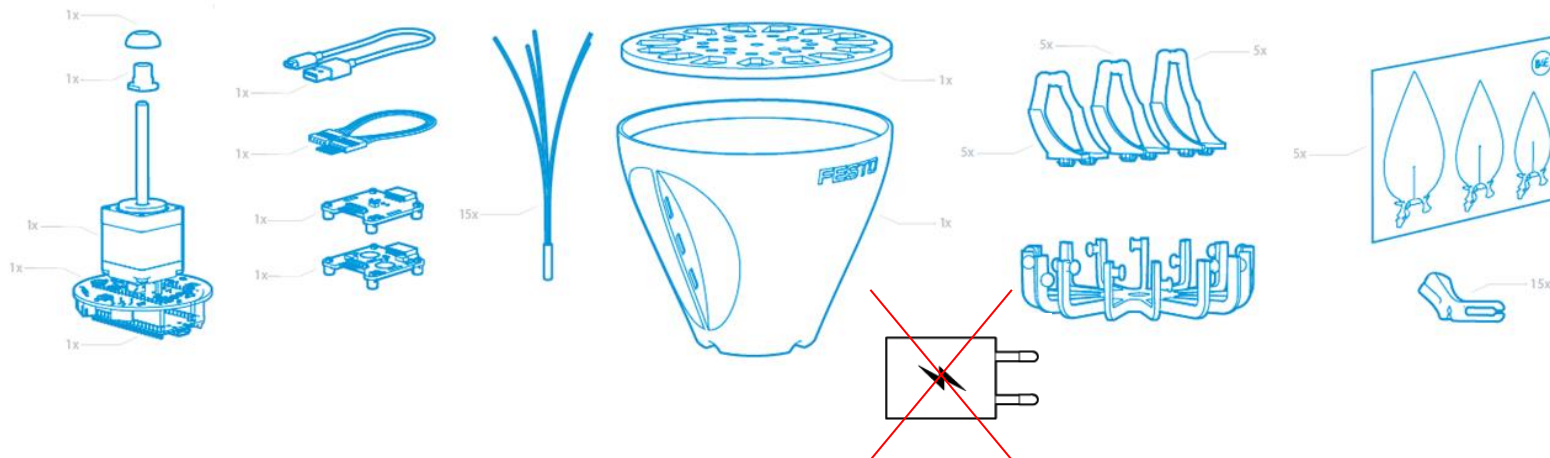
# Bionics4Education – The Bionic Flower

## What is the Bionic Flower?

The Bionic Flower is a sensor-controlled electronic flower developed by Festo Didactic. You can do several experiments with the Bionic Flower and discover biological phenomenas as well as technical issues. For example, you will work with sensors or a microcontroller and get to know how they can influence the petal movements.



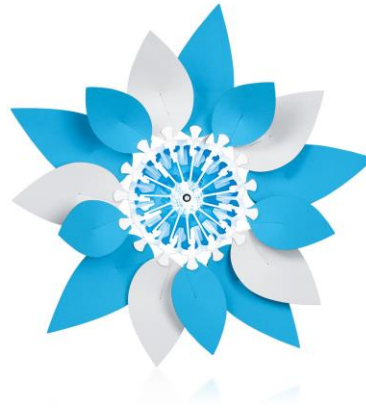
## Experimentalset: Hardware Components



## Bionics4Education – The Bionic Flower

### A robot flower inspired by the plant world

- 🌸 Festo Didactic took the mechanisms of action of water lilies and mimosa as a model for the development of the Bionic Flower.
- 🌸 These plants have one thing in common: the opening and closing of their petals or leaves due to external stimuli. The effects serve for reproduction and protection against natural enemies.



# Bionics4Education – The Bionic Flower

## Biological Background Information

### Natural Role Model – Water Lilies

The water lily is an aquatic plant that is anchored in bodies of water. The single standing flowers have a spiraled-like structure triggered by light and thus allow the opening and closing of their petals. Day-flowering water lilies open their petals in the morning and thus attract insects to pollinate the flowers. But how exactly does the movement of the petals work?

The principle is based on differences in growth between the outside and inside of the petals. So, the inside of the petals grows under bright light and the flower opens. If there is not enough light, the outer side grows and the flower closes.



### Natural Role Model – Mimosa Plant

The leaves of the mimosa plant fold one after the other when subjected to mechanical stimuli such as touch or vibration nearby, and the stem sinks. This serves to protect against possible predators.

The successive folding of the pinnules takes place like a chain reaction (domino effect). An ingenious mechanism within the water-filled cell structures in the pinnules ensures that water is released from the cell when an external stimulus is applied. This means that no more water presses from the inside against the cell walls, the cell relaxes and the pinnules contract.



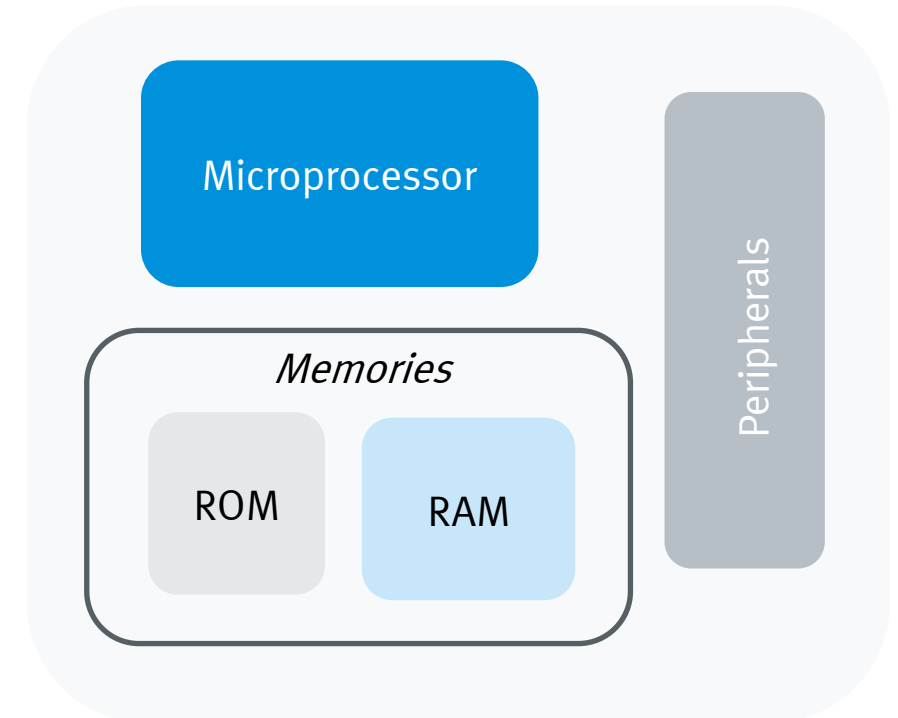
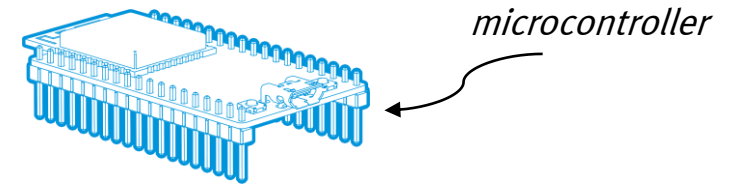
# The Bionic Flower – Hardware

## Theory Part . Electronic component- Microcontroller

The BionicFlower has controlled with one ESP32 microcontroller. Microcontrollers are omnipresent in our daily lives (homes appliances, robotic,..). A microcontroller is a composed of 4 parts :

- **microprocessor** : it is a component that allows to execute the programs embedded in the microcontroller to support the information processing and sending commands part.
- **programmable memory** (ROM : Read Only Memory) : this memory contains the instructions of the program.
- **data memory** (RAM : Random Access Memory) :  
in this memory will be stored the temporary data necessary for the calculations.
- **peripherals** : some ports are available to interact with outside like sensors, actuators, display, etc. Thus, the microcontroller can receive data but also send commands.

So, the microcontroller is capable to execute a program, process information and communicate with other components. The ESP32 allows to control the degree of opening of the flower and the color of the LEDs.



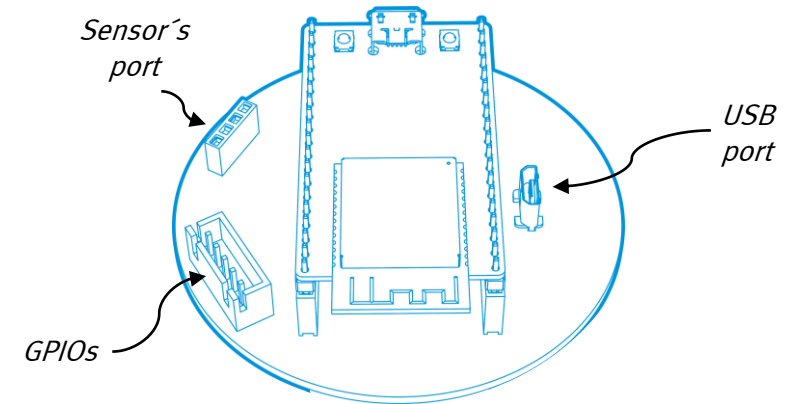
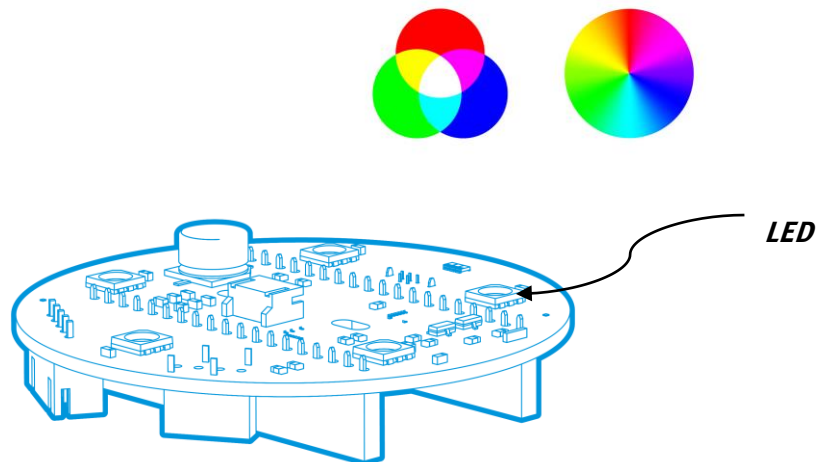


# The Bionic Flower – Hardware

## Theory Part. Electronic component- PCB

Festo teams developed a printed circuit board (PCB) to assemble all the necessary components for the good working for the BionicFlower.

On this PCB there are 5 **LEDs**. Each LED is an RGB LED, this means that each LED is the combination of three colors (Red, Green, Blue): thanks to these three colors all colors can be created.



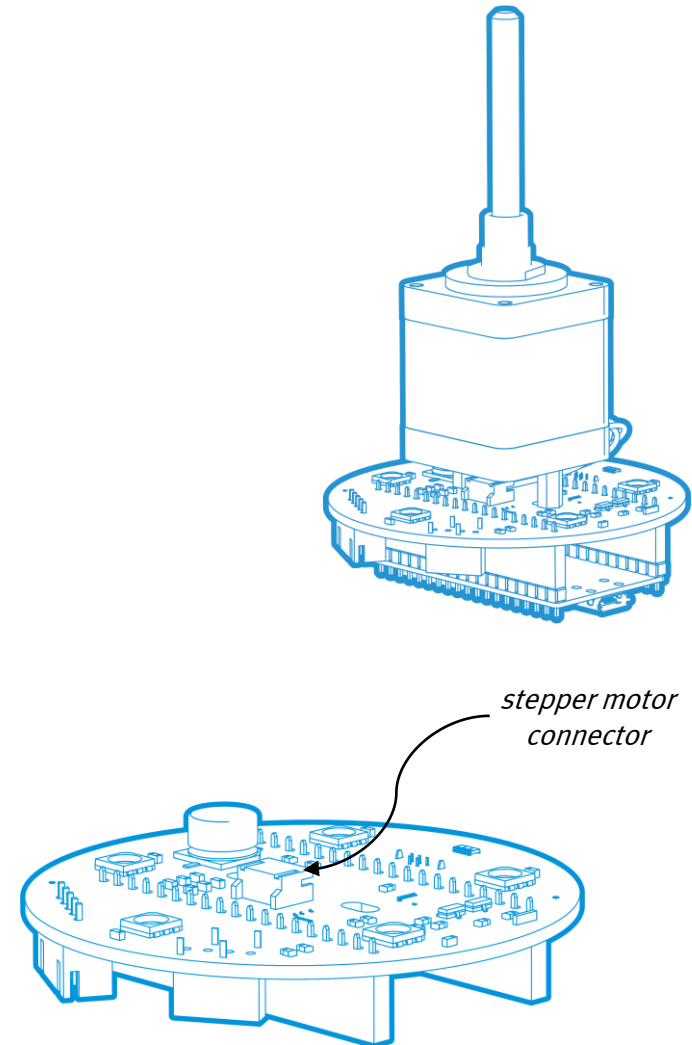
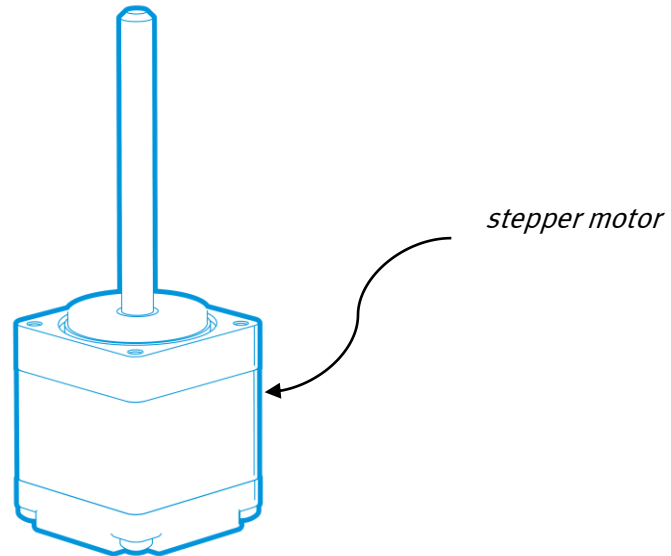
On the other side of the PCB, there are :

- **USB port** : to powered and/or upload your program on the microcontroller.
- **Sensor's port** : it is a I<sup>2</sup>C port, it is to connect the BionicFlower's sensors.
- **GPIOs** : some GPIO (General Purpose Input/Output) are available.

# The Bionic Flower – Hardware

## Theory Part. Electronic component- Stepper motor

A stepper motor allows to convert an electric impulse in angular motion. In the BionicFlower, the stepper motor is associated with a screw which allows to convert the angular motion in on a translational movement. This provide to open or close the BionicFlower.



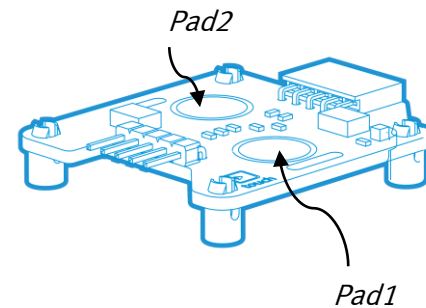


# The Bionic Flower – Hardware

## Theory Part. Electronic component- Sensors

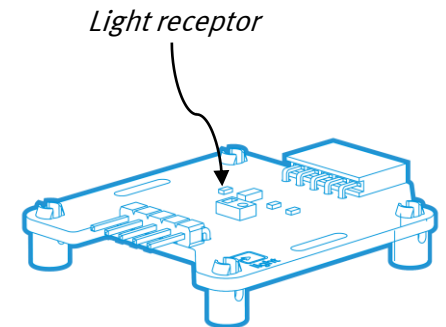
### *Touch sensor*

The touch sensor is composed of two pads : Pad1 and Pad2. The sensor detects if one of pads or both pads are pressed. This sensor sends boolean data : true or false.



### *Light sensor*

The light sensor allows to measure the light intensity in lux . Lux is the unit of measurement used to measure the luminous intensity received per unit area.



# Introduction - Programming

## What is programming ?

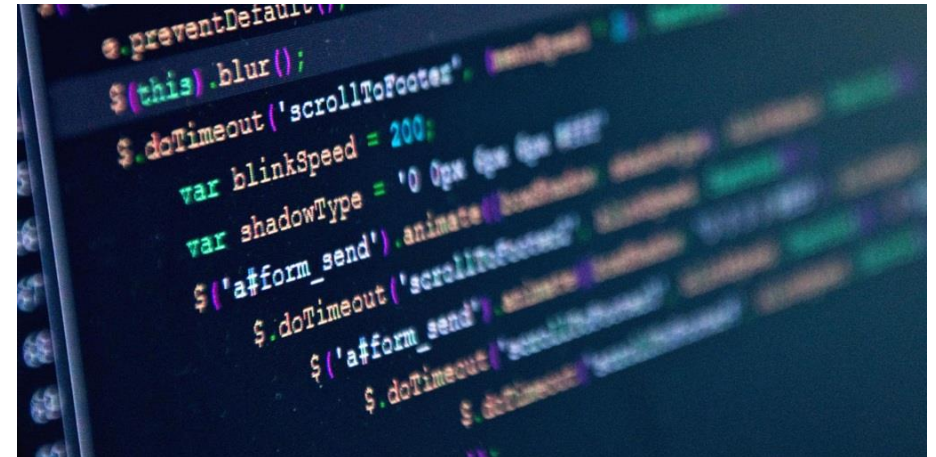
Programming is the process of writing computer programs. A program is a sequence of instructions that the computer must carry out.

To communicate with the computer and thus write programs, a **programming language** is used. Indeed, humans cannot understand machine language, it is far too complex and illegible, it is the **binary program**. For this we have created different languages. The programming language allows to create the **source code** : program written and comprehensible by humans. Thus, the same program can be written in different languages (Java Script, Python, C, PHP,..). However, depending on the application, some languages are more suitable. For example, the C and C++ languages are the programming languages most used in robotics.

Programming steps are :

- **design** : determination of the objective of the program and analysis of the functional aspect of the program with in particular the development of an algorithm or the analysis of the data inputs/outputs.
- **implementation** : coding of the program with a choice of programming language, this transforms the algorithm into source code.
- **compilation** : translation of source code into binary code.

Once the program is compiled, the binary file will be sent to the microcontroller via USB port. *Open Roberta* can be used to write the code, compiled and upload to the Bionic Flower.




# Introduction - Programming

## Theory Part. Open Roberta tool

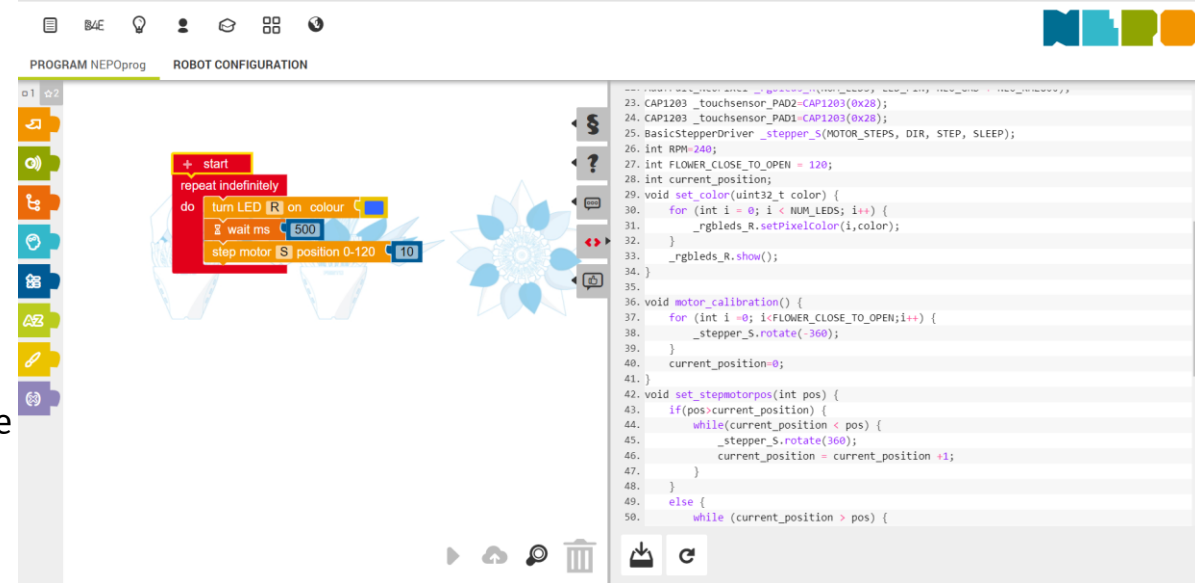
**OpenRoberta** is a web programming tool and provides programm blocks to give instruction to some robots including the BionicFlower.

The blockly language allows to learn easy programming. Indeed, blocks make it possible not to worry about the indentation, the syntax, type of data, etc. of the program. When a block is used, it is transcribed in simplified C language.

If you are curious, you can see this source code by clicking on this item: 



```
void set_color(uint32_t color) {
    for (int i = 0; i < NUM_LEDS; i++) {
        _rgbleds_R.setPixelColor(i,color);
    }
    _rgbleds_R.show();
}
```



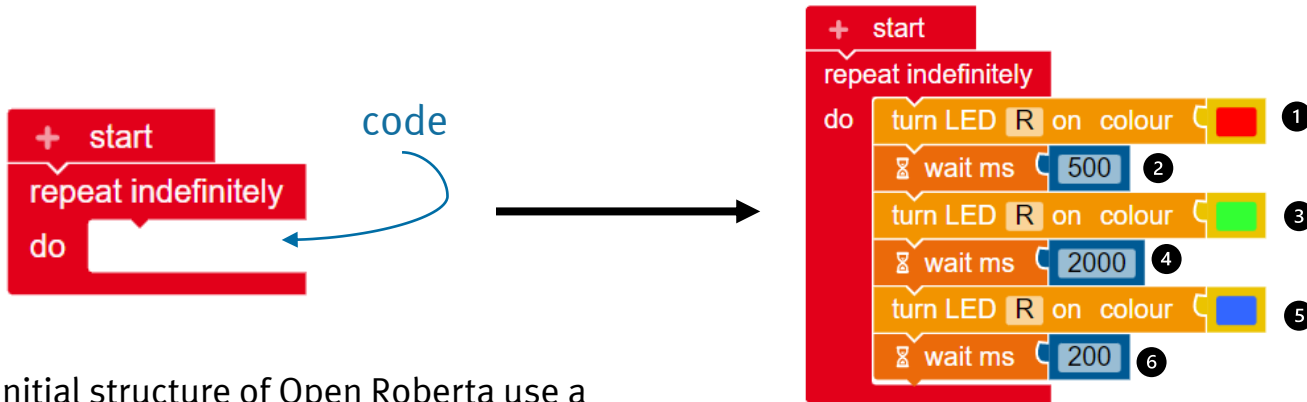
Open Roberta Lab : <https://lab.open-roberta.org/>

To create your program, you will be able to use different instructions in block form located on the left of the screen.

In robot configuration menu, you can declare which hardware is used in your program : LEDs and stepper motor are always declared, but you can add sensors : touch and light sensor .

# Introduction - Programming

## Theory Part. Instructions : structure on OpenRoberta Lab



The initial structure of Open Roberta use a “indefinitely” loop. That means that the code will repeat as long as the microcontroller is switched on.

The microcontroller executes instructions always in the same order. The highest instruction will be executed first, then the next and so on. Thus, for this example the instruction 1 will be executed first, then the 2, etc.

In addition, with the indefinitely loop, as long as the microcontroller has power, the execution sequence is 1,2,3,4,5,6,1,2,3,4,5,6,1,2,etc.

# Introduction - Programming

## Theory Part. Instructions: Wait

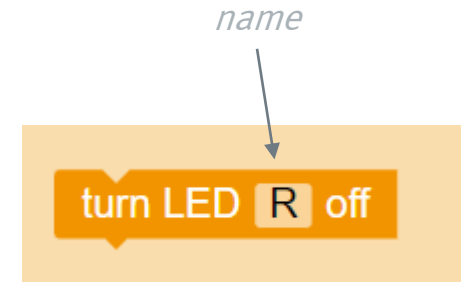
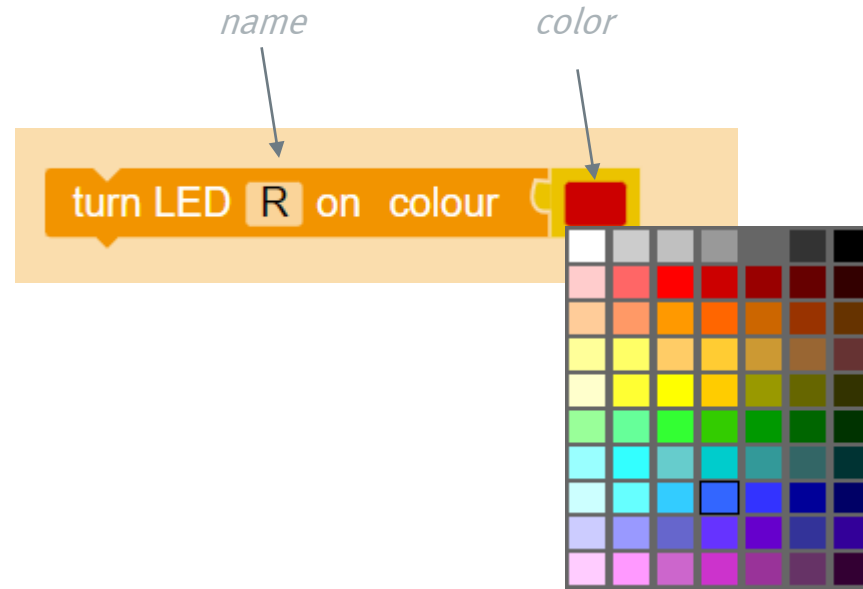
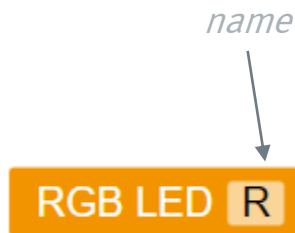


The *wait* instruction allows to put a waiting time between two instructions. This instruction needs to have a parameter (in blue). It is a number which determines the delay in milliseconds.

# Introduction - Programming

## Theory Part. Function for Bionic Flower: turn LED and turn off LED

In configuration robot, you can find the object „ RGB LED“, the name of this object is R by default, but you can change the name. It is not possible to control LEDs one by one, this object represents all 5 LEDs.



The **turn LED off** instruction allows to turn off the 5 LEDs.

The **turn LED R on color** instruction allows to turn on the 5 LEDs in the color which is selected with the color panel. There are 70 colors.



# Introduction - Programming

## Theory Part. Function for Bionic Flower: step Motor

In configuration robot, you can find the object „step motor“, the name of this object is S by default, but you can change the name.

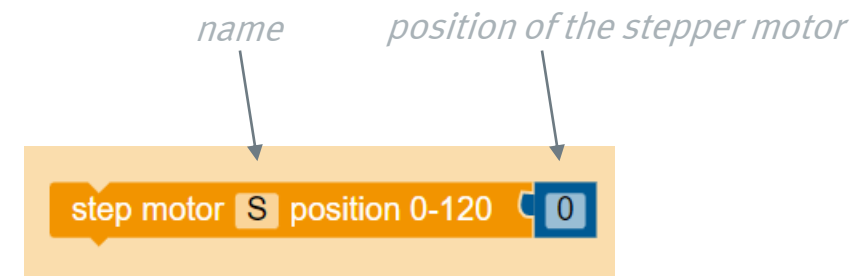


The *step motor* instruction allows to move the stepper motor.

To move from fully closed to fully opened, the stepper motor needs to do 120 turns.

This stepper motor instruction needs to have a parameter (in blue). It is a number which gives the number of turns to determinate the position of the opening of the flower. To completely close the Bionic Flower, this number is 0. To completely open the Bionic Flower, this number is 120.

**!** The setting of the motor position corresponds to the current position of the motor, it will not move.



# Introduction - Programming

## Theory Part. Variable

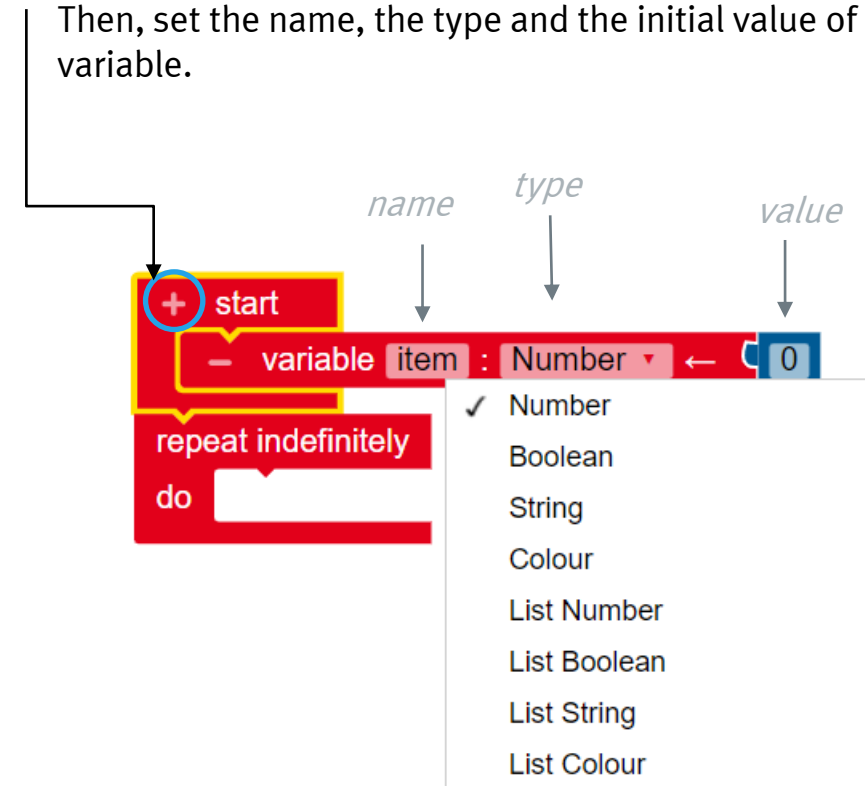
Variables are used to store data.

Each **variable** is defined by 3 attributes :

- **name** : the name of a variable can be any string of characters. All names are possible like “toto” for example. However, to make it easier to understand the code, it is customary, most often, to use a name that describes the variable. For example, if this variable must store data from a proximity sensor, then we can call it “distance.”
- **type** : the type of the variable defines what type of data will be stored in this variable. This allows the best organization of the memory. The most used types are number, Boolean (true or false) and string or characters.
- **value** : the value of the variable store the data. For example, the value of variable *distance* it could be 10 for 10 cm.

To create a variable on Open Roberta, you must click on the add symbol next to the “start”.

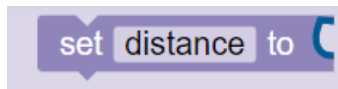
Then, set the name, the type and the initial value of the variable.



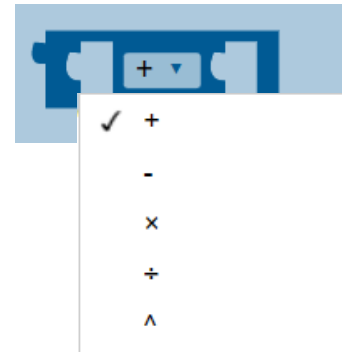
# Introduction - Programming

## Theory Part. Variable & Math Operation

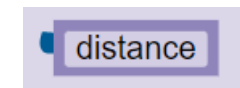
Once the variable is created, it is possible to change it using the set block to statement in the code. This instruction requires a parameter of the same type as the type declared for the variable. Thus, after this instruction, the variable will be set to the value of the parameter.



The following block allows to put an integer as parameter.



To change the values of the variables or for the conditions, we have the possibility to use mathematical operations such as addition, subtraction, multiplication, division and power. In addition, this instruction needs two parameters, both is integer.

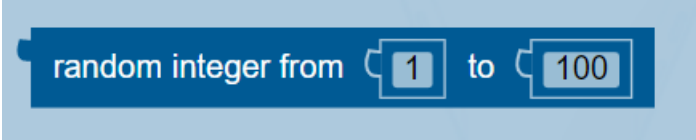


In addition, it is possible to use the value of a variable using this instruction.

# Introduction - Programming

## Theory Part. Random

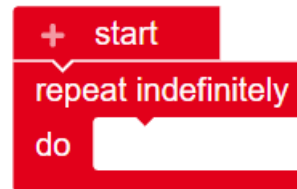
The random instruction allows to generate a random number between the first and the second number included. The following example generates a random number between 1 and 100.

A screenshot of a programming instruction. The text 'random integer from' is followed by a blue box containing the number '1', then the word 'to', and another blue box containing the number '100'. The entire instruction is set against a light blue background.

```
random integer from 1 to 100
```

# Introduction - Programming

## Theory Part. Loop: repeat indefinitely



The structure **repeat indefinitely** is a loop that allows to repeat indefinitely the instructions placed in that loop. You never get out of this loop. That is why there is no exit condition.

# Introduction - Programming

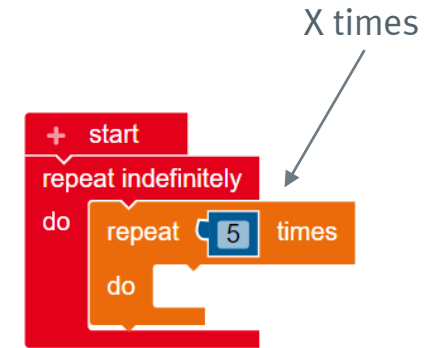
## Theory Part. Loop: repeat X times

The **X times loop** allows to execute instructions X times.

That means that the instructions which are included on the loop's structure (instructions A) will be executed X times.

This loops requires a positive number , X. This number is the number of times the instructions will be executed. After that, the code will continue the following instructions (instructions B).

```
repeat X times  
{  
  /*instructions A*/  
}  
/*instructions B*/
```

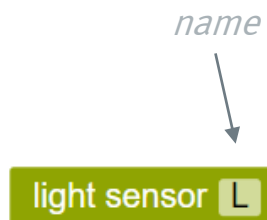




# Introduction - Programming

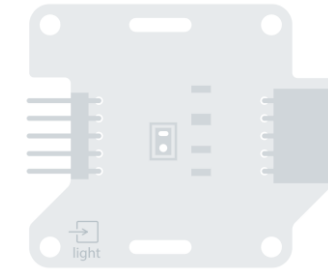
## Theory Part. Variable: Get Value from light sensor

In robot configuration, you can add the light sensor. You can change the name. Its name is L by default.



```
get value lx light sensor L
```

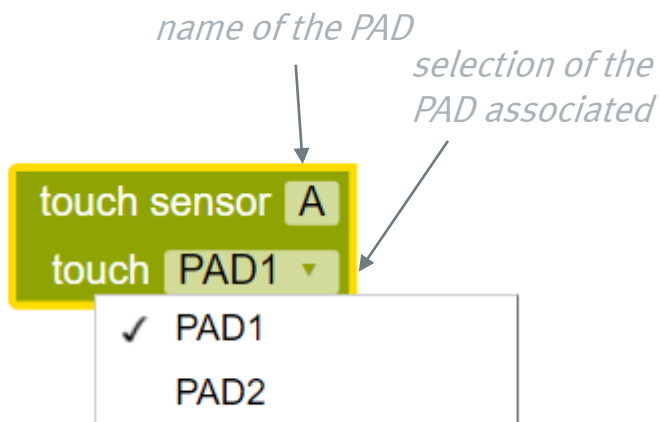
The light sensor block returns the data from the light sensor, that means the value of the luminosity in lux.



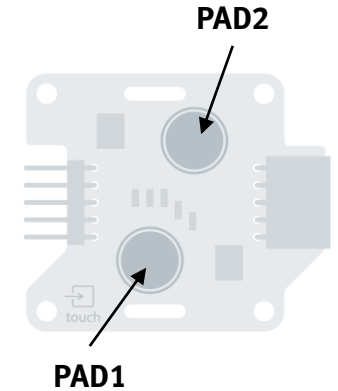
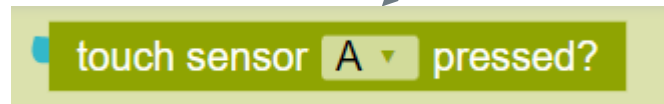
# Introduction - Programming

## Theory Part. Variable: Get Value from touch sensor

In robot configuration, you can add the touch sensor. You can change the name of the PAD. Each PAD must have a unique name.



*selection of the PAD with the PAD's name*



The touch sensor block returns a boolean (True or False). This boolean is True if the PAD selected is pressed, otherwise False.

# Introduction - Programming

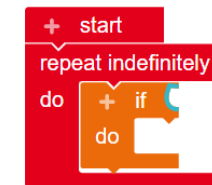
## Theory Part. Loop: if do / if do else / If do else if else if

The *if structures* are very used in programming. Indeed, these allow to create new behavior of your code based on parameters such as time, the value of a variable, etc.

If structures are based on the following principle: a condition is tested, if it returns true then the program executes the instruction contained in the body associated with that condition, otherwise the code ignores the instructions and continues the code.

- The **if do** structure is the simplest if structures. Indeed, the condition is tested. If this condition returns true, instructions in the if's body are executed (instructions A), then the program continue and execute the following instructions (instructions B). Otherwise, if the condition is false, the program ignore the instructions on the if structure and execute the following instructions.

```
if (condition 1)
{
  /*instructions A*/
}
```



*if do structure*

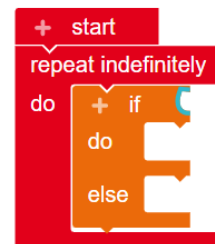
# Introduction - Programming

## Theory Part. Loop: if do / if do else / If do else if else if

- The **if do else** structure allows to separate 2 cases : the condition is true and the condition is false. And depending on this result, it there will be two different behavior. Indeed, the condition is tested. If this condition returns true, instructions in the if's body associated with the condition are executed (instructions A), next the program continues and execute the following instructions. Otherwise, the condition is false, the program executes the instructions on the else body (instructions B) and then the program continues to execute the following instructions.

```

if (condition 1)
{
  /*instructions A */
}
else
{
  /*instructions B */
}
    
```



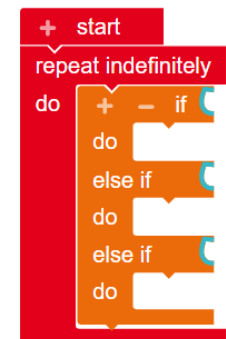
*If do else structure*

- The **if do else if** structure allows you to separate several cases. The conditions will be tested one after the other until the condition returns true. Then the instructions associated with this condition will be executed. Then all other conditions will be ignored. The program exits this if structure and executes the following instructions.

For example, if the condition1 test return false, the condition2 will be tested. If this condition return true, the instructions B will be executed, then the other conditions will be ignored. If the condition2 return false, the following condition will be tested and so on. If no condition returned true, then none of the instructions in the if structure were executed.

```

if (condition 1)
{
  /*instructions A */
}
else if (condition 2)
{
  /*instructions B */
}
...
else if
{
  /*instructions N */
}
    
```



*If do else if structure*

# Introduction - Programming

## Theory Part. Loop: repeat while

The **while loop** allows to execute instructions while that the condition associated is true.

That means that for each turn of the loop, the condition is tested. If the condition returns true, the instructions included in the loop's body are executed (instructions A). Otherwise, that means that the condition test returns false, code exits the loop and executes the following instructions (instructions B).

```
repeat while (condition )  
{  
    /*instructions A*/  
}  
/*instructions B*/
```



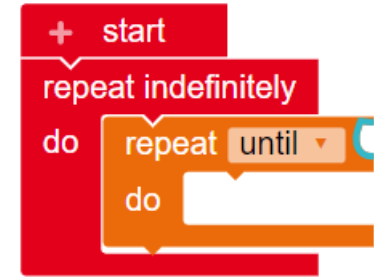
# Introduction - Programming

## Theory Part. Loop: repeat until

The **until loop** allows to execute instructions until that the condition associated becomes true.

That means that for each turn of the loop, the condition is tested. If the condition returns false, the instructions included in the loop's body are executed (instructions A). Otherwise, that means that the condition test returns true, code exits the loop and executes the following instructions (instructions B).

```
repeat until (condition )  
{  
    /*instructions A*/  
}  
/*instructions B*/
```





# Introduction - Programming

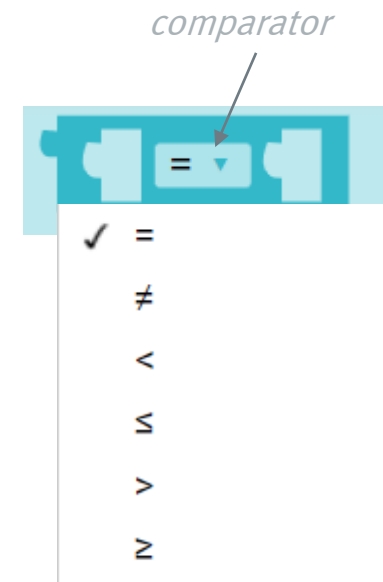
## Theory Part. Loop: condition/ comparaison

Conditions are important because they allow to introduce structures such as if structures, while loops or until loops.

Testing a condition returns a Boolean : **true** or **false** .

If you want to compare two values of two variables of type number, we can use the comparison structure. To compare two numbers, you can use the following comparators : equality, non-equality, inferior, inferior or equal, superior , and superior or equal.

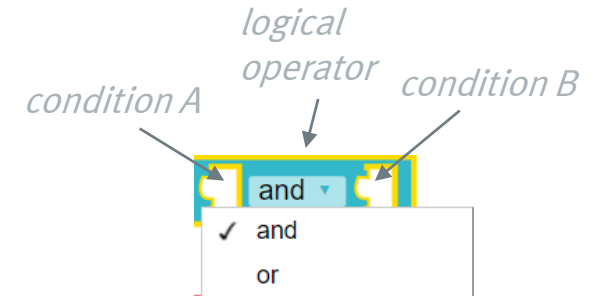
This block requires one comparator, and two number. It can be a block-number or a variable with a number type.



# Introduction - Programming

## Theory Part. Loop: condition / AND OR logical operator

To make conditions more complex, you can use the logical operator **AND** and **OR**. This logical operator allows to bring together two condition.



The **AND** operator requires both conditions to be true to return true.

### AND

Condition A	Condition B	Result
False	False	False
True	False	False
False	True	False
True	True	True

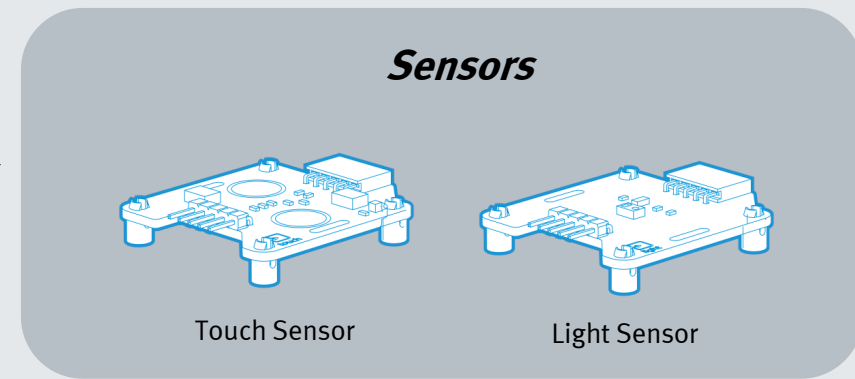
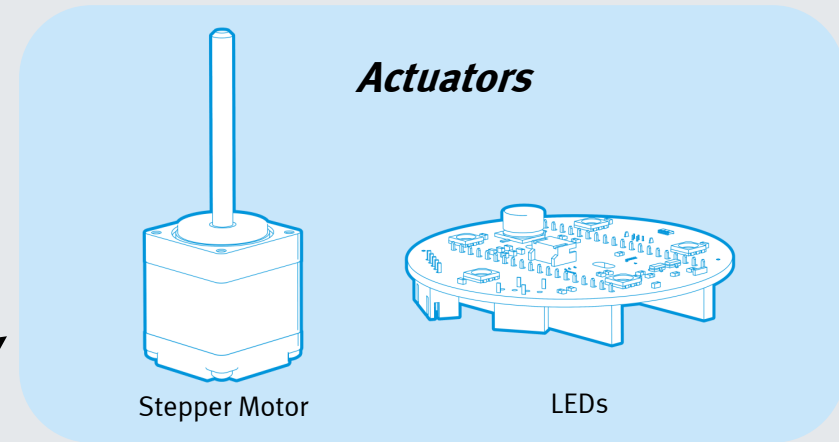
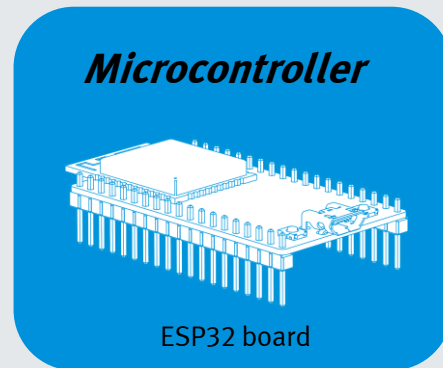
The **OR** operator requires at least one condition to be true to return true.

### OR

Condition A	Condition B	Result
False	False	False
True	False	True
False	True	True
True	True	True

# Introduction – Open Roberta

Programming the Bionic Flower  
Process with Open Roberta

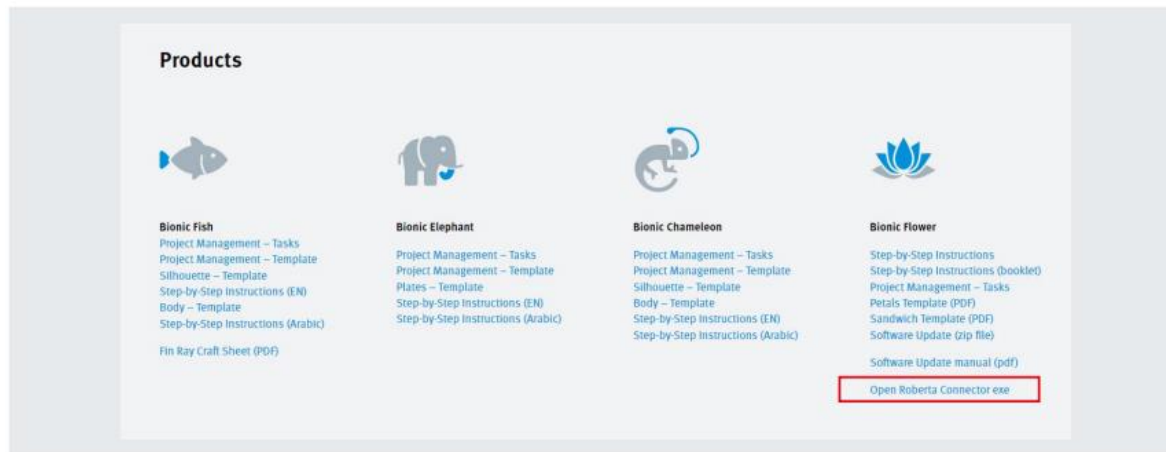


# The Bionic Flower – Open Roberta

## How to install connector of Open Roberta ?

### *Download and Unzip the connector*

- Go to Bionics4Education web site : <https://www.bionics4education.com/startseite/support>
- Download the Open Roberta Connector.exe file



- Unzip the file

Do you prefer a video to explain how to install Open Roberta?

Go to the YouTube video with the following link :  
<https://www.youtube.com/watch?v=JNCAL6aCfbl>

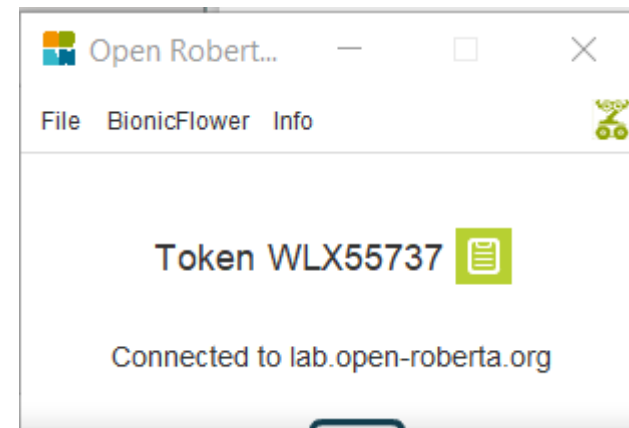
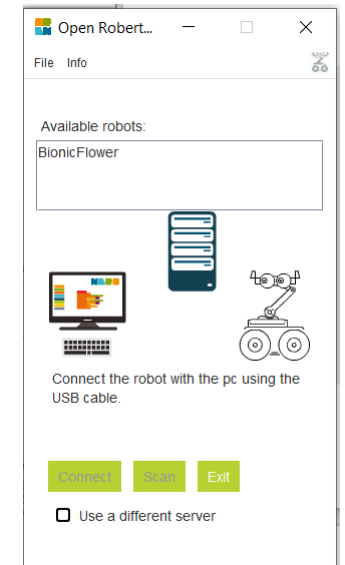
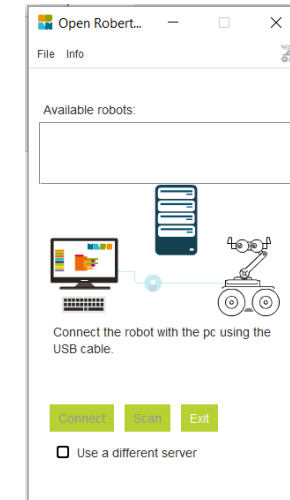
# The Bionic Flower – Open Roberta

## How to install connector of Open Roberta ?

### Setup Open Roberta Connector

- Open the Open Roberta Connector
- Connect the Bionic Flower on your PC with the USB cable
- Wait that the BionicFlower appear
- Click on connect
- Copy and Paste the Token

Name	Date modified	Type
Drivers	5/24/2022 12:18 PM	File folder
libs	5/24/2022 12:18 PM	File folder
resources	5/24/2022 12:18 PM	File folder
OpenRobertaConnector.exe	3/28/2022 1:42 PM	Application



# The Bionic Flower – Open Roberta

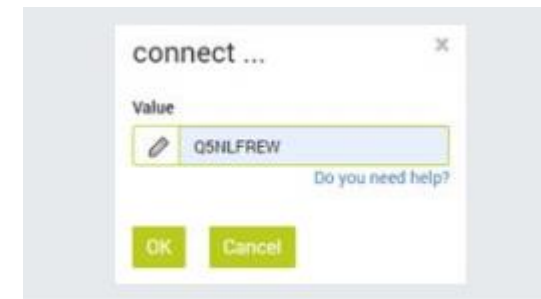
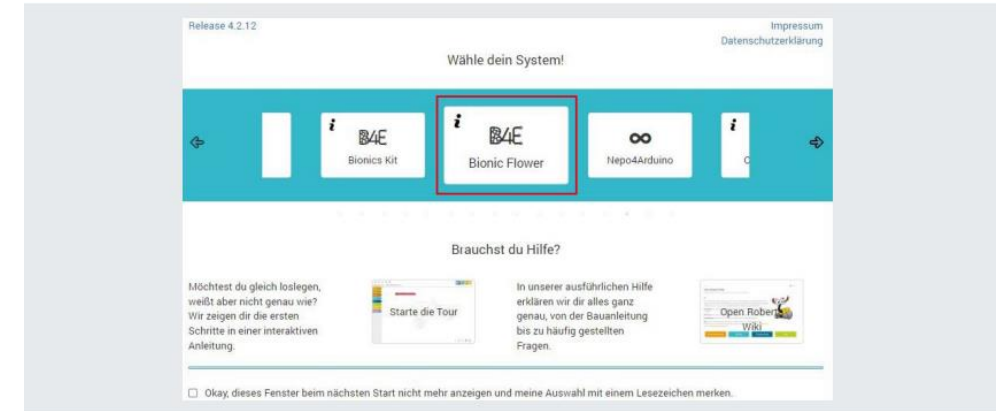
## How to install connector of Open Roberta ?

### Setup your Bionic Flower on Open Roberta

- Go to Open Roberta web site: <https://lab.open-roberta.org/>
- Select the Bionic Flower as your system
- Go to the B4E menu and click to connect the Bionic Flower



- Past the token to connect the Bionic Flower and click on “Ok” button

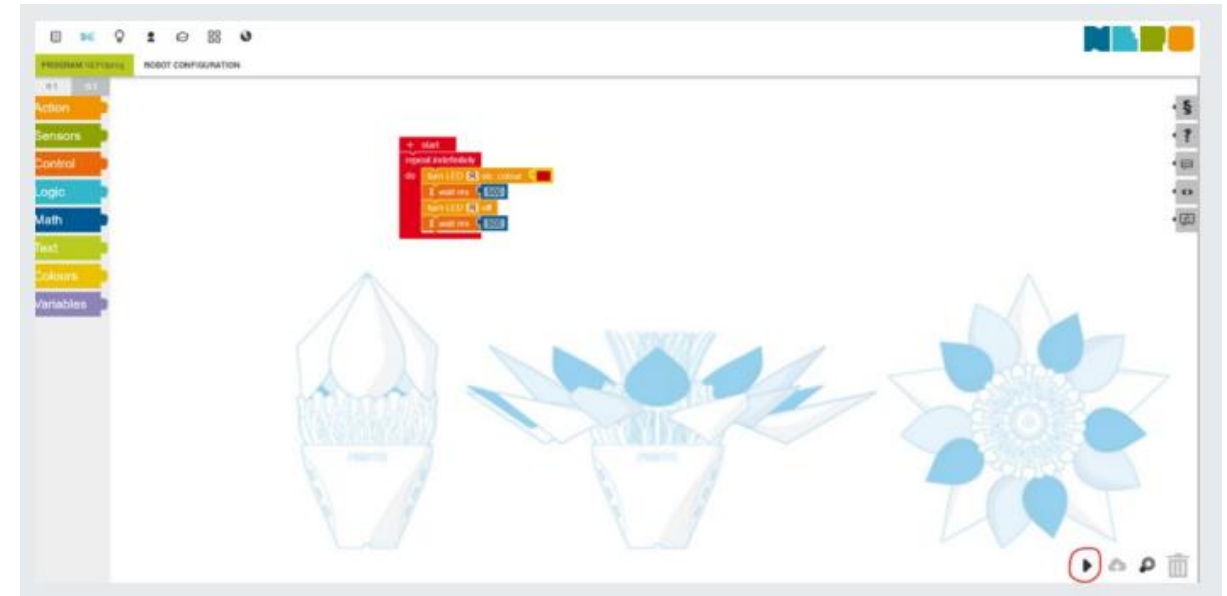


# The Bionic Flower – Open Roberta

## How to install connector of Open Roberta ?

### *Upload a code on your Bionic Flower*

- Do your first code
- To upload your code on your BionicFlower, click on the on arrow at the bottom right
- The B4E logo blink to indicate that the code is uploading on the Bionic Flower



# The Bionic Flower – Open Roberta

## How to install connector of Open Roberta ?

### *Import/Export/Reset*

- Import

Go to the first menu and click on the export program A file with format .xml is created on your download folder

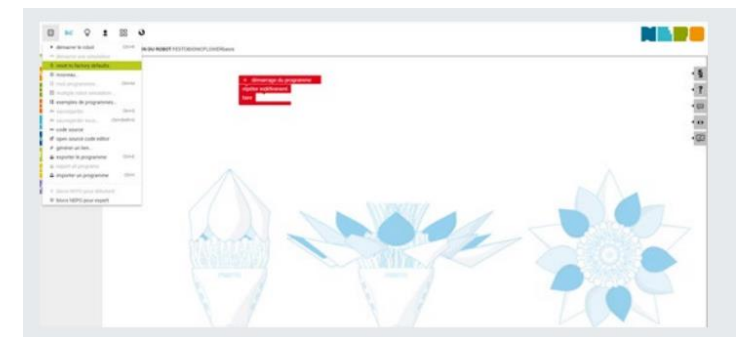
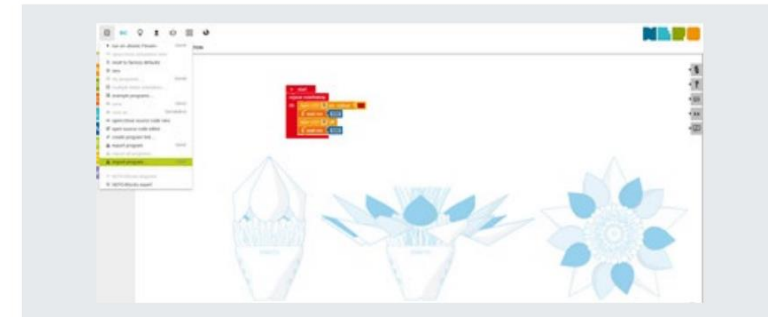
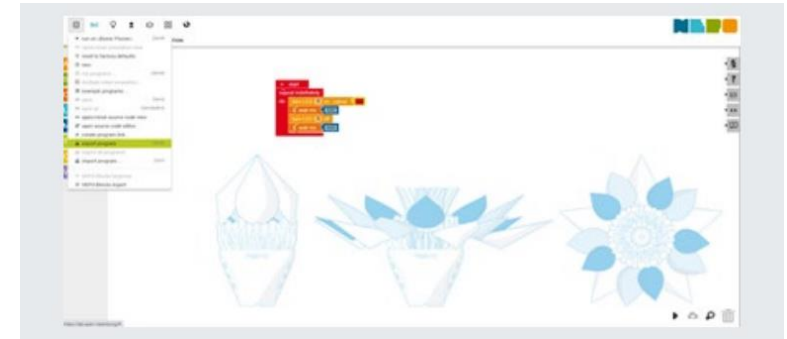
- Export

Go to the first menu and click on the import program Your code appears on the Open Roberta interface.

- Reset default code Bionic Flower

If you want to reset the Bionic Flower with the initial code, go to the first menu and click on the “reset to factory defaults”.

*Exit:* Before exit Open Roberta, think to import your code if you want to use it for the next time. To exit the Open Roberta, on the Open Roberta Connector disconnect the Flower and disconnect your flower to your computer.





## Coding Projects - Overview

With using *Open Roberta*, you will write programs to interact with the flower and create projects. To help you to do it, three projects are already done, and we will see together, step by step how to program and create a story with your Bionic Flower.

**Each DIY projects match** with a different level of programming.

- Level 1 : **Carnivorous Plant**

You will recreate the incredible ability of carnivorous plants to capture insects.

Initially, the flower is open and lights in green. If you press the touch sensor (simulating the insect) the flower closes and is blinking in red. After a delay, the flower opens again and lights up in green.

- Level 2 : **Day of the flower**

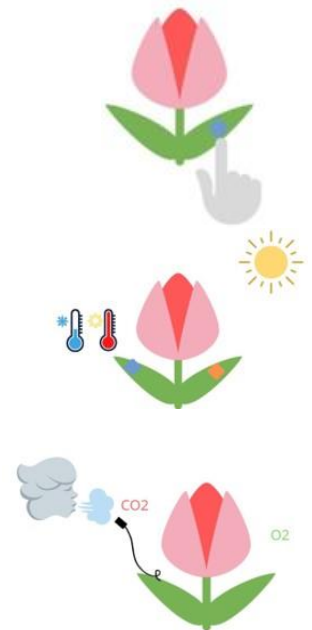
You will recreate a full day of a waterlily flower. During the day, the flower opens and closes at night.

Depending on the luminosity, the flower changes the LEDs color and open or close.

- Level 3 : **Photosynthesis**

You will recreate the photosynthesis. The photosynthesis process needs, among others, chlorophyll's cells and solar energy.

With the touch sensor, you will select the right color for the photosynthesis : green for the chlorophyll's cells. If the luminosity is enough and that the chlorophyll's cells are present, so the flower will open and change on a new color, this translates the fact that all requirements for the process are present.



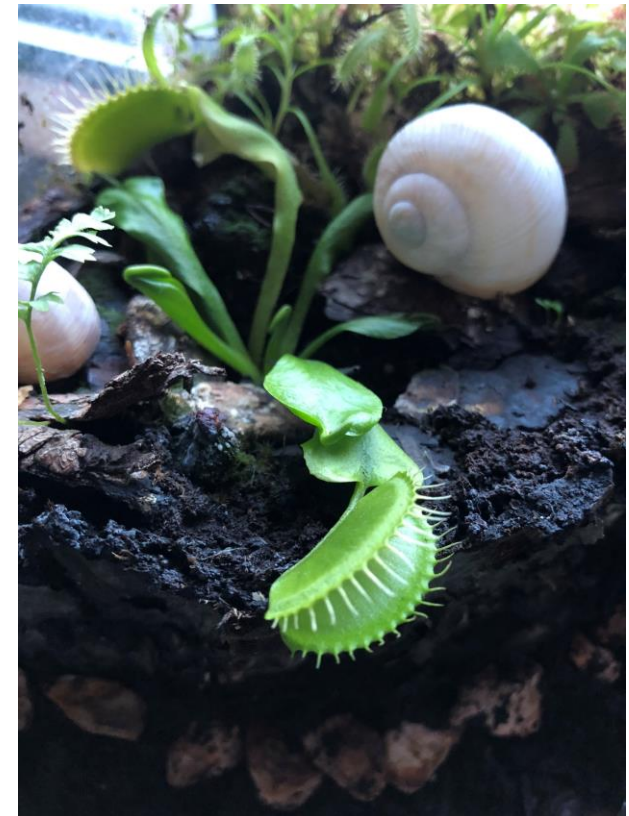
# Coding Project – Carnivorous Plant

## Biological background information

Carnivorous plants are unique in their ability to attract, capture and assimilate (in whole or in part) their prey, which are mostly insects. Some species can even digest small frogs or rodents, such as mice.

Carnivorous plant's traps are defined as “active” if they are mobile and fast. Indeed, when the carnivorous plant detects an insect, it immediately closes and traps the prey. After digestion, the plant reopens while waiting for the next prey.

In the next step you will write a program to interact with the Bionic Flower. Initially, the Bionic Flower is open and lights in green. If you touch the touch sensor (simulating the prey) the flower fastly closes and is flashing in red. After a delay, the flower opens again and lights up in green. In addition, you will also be able to improve this scenario by using the gesture sensor. You will be able to use the 8 movements of the gesture sensor to define different behaviors of the flower. For example, the "forward" movement to trigger the closing of the flower and the flash of the LEDs. The movement "back" to reopen the flower. The movements "right" and "left" to change the color of the flower. The "up" and "down" movements to modify the intensity of the color. And "clockwise" and "anti-clockwise" movements to create a charging movement with the LEDs respectively clockwise or anti-clockwise.



# Coding Project – Carnivorous Plant

## Coding activities

### ***STEP 1*** : *Flashing of the LED*

Objective: Create a flashing of the LED.

Tips: Think to add delays between turn off and turn on the LED.

### ***STEP 2*** : Touch sensor implementation

Objective: Create a danger situation with the LED.

The flower is green. And if a touch is detected, the flower blink in red.

Tips: Think to add the touch sensor to the bionic flower.

### ***STEP 3*** : Using of the motor

Objective: Create a danger situation the LED and the motor.

The flower is green and opened. And if a touch is detected, the flower is in red and the flower closes. After a delay, the flower opens again.

### ***STEP 4*** : Create a danger situation for the carnivorous plant

Objective: Create a danger situation the flashing with the LED and the motor.

The flower is green and opened. And if a touch is detected, the flower is flashing in red and the flower closes. After a delay, the flower opens again.

Tips: Use a variable “Step\_motor” to store the number of the step of the motor and use a math function to increase or decrease it.

# Coding Project – Day of the Flower

## Biological background information

Water lilies are aquatic plants that are anchored in bodies of water. The single standing flowers have a spiraled-like structure triggered by light and thus allow the opening and closing of their petals. Day-flowering water lilies open their petals in the morning and thus attract insects to pollinate the flowers.

The principle is based on differences in growth between the outside and inside of the petals. So, the inside of the petals grows under bright light and the flower opens. If there is not enough light, the outer side grows and the flower closes.

In this project, you will write a program to interact with the flower and to recreate the day of a flower (e.g. water lilies). The flower's color depends on the time of day (morning (ambient luminosity), day (illuminate with the light of your smartphone), or night (put your hand on the sensor)). The flower needs light to open (it is open when it is day or morning and closed when it is night). In addition, the degree of opening of the flower will depend on the temperature. And a high temperature (temperature sensor in your hands) improves the growth of the flower, so the flower opens more.



During the night: Water lilies are closed



During the day: Water lilies are open

# Coding Project – Day of the Flower

## Coding activities

### **STEP 1 :** Turn on the LED

Objective: Turn on the LED in blue and yellow.

Tips: Think to add delays (1000 ms for example) between each color change so that you can see its changes, otherwise it would be too fast for your eyes

### **STEP 2 :** Light sensor implementation

Objective: Define the day and the night depending on the luminosity value.

Store the brightness value in a variable. If the luminosity is higher than 10, it is the day, the flower must then light up in yellow. Otherwise, it is night, and the flower should light up blue.

Tips: think to add the light sensor to the bionic flower

### **STEP 3 :** Using of the motor

Objective: Open and close the motor.

Store the brightness value in a variable. If the luminosity is higher than 10, it is the day, the flower must then light up in yellow and the flower opens. Otherwise, it is night, and the flower should light up blue and the flower closes.

### **STEP 4 :** Create the day of the flower

Objective: Create the day of the flower.

The flower is opened on the day and is closed on the night. Store the brightness value in a variable. If the luminosity is higher than 10, it is the day, the flower must then light up in yellow and the motor opens step by step. Otherwise, it is night, and the flower should light up blue and the motor closes step by step.

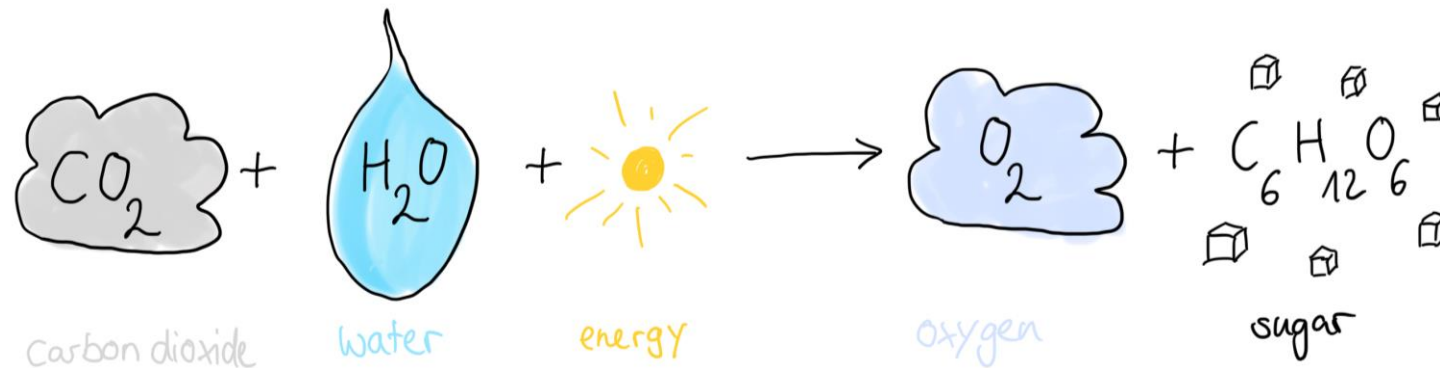
Tips: Use a variable “Step\_motor” to store the number of the step of the motor and use a math function to increase or decrease it.

# Coding Project – Photosynthesis

## Biological background information

The photosynthesis is one of the most important biochemical processes in nature. Without this phenomena, there would be no life on our earth.

In the process of photosynthesis, carbon dioxide (CO<sub>2</sub>) and water (H<sub>2</sub>O) are converted into oxygen (O<sub>2</sub>) and sugar (C<sub>6</sub>H<sub>12</sub>O<sub>6</sub>) with the help of energy (sunlight). The sugar is used by the plants to live and grow. The oxygen they do not need is released into their environment, which in turn is needed by other living organisms to survive.



In this project, you will recreate the effect of photosynthesis. You will write a program to interact with the flower. You will select “green” (which represents chlorophyll cells) with the touch sensor. The light sensor will allow you to define if it's day or night. The flower opens if it's day and closes if not. In addition, you can blow on the CO<sub>2</sub> sensor to represent the presence of CO<sub>2</sub>.

# Coding Project – Photosynthesis

## Coding activities

### **STEP 1 :** Turn on a random color

Objective: Turn on the LED in random color.

Define a random color between 4 colors (red, green, blue, and yellow) and turn on the LED in this color.

Tips: Use the random function to have a random number between 1 and 4.

### **STEP 2 :** Touch sensor implementation

Objective: Using the touch sensor for change the color of the flower.

Define a random color between 4 colors (red , green, blue, and yellow) and turn on the LED in this color. In indefinitely way, if a touch is detected, the color change for the next color.

Tips: Think to add the touch sensor to the bionic flower. Use a variable “Color” to store the number of the color and use a math function to increase or decrease it.

### **STEP 3 :** Light sensor implementation

Objective: Define the day and the night depending on the luminosity value.

Store the brightness value in a variable. If the luminosity is higher than 10, it is the day. Otherwise, it is night. If it is the day and the color is green, so the flower is blinking 5 times with a new color (purple). After a delay, the flower lights up in random color again.

Tips: Think to add the light sensor to the bionic flower

### **STEP 4 :** Create the photosynthesis

Objective: Create the photosynthesis.

You need to have a chlorophyll cells and a light to that the flower growth well. Define a random color. You can change the color with the touch sensor. If the color is green and the luminosity is enough, the flower opens and turn on a new color. And after a delay, the flower closes and lights up in a new random color



# Finished!

## Bionics4Education: Bionic Flower Courseware for Open Roberta

Congratulations – you have finished this course

