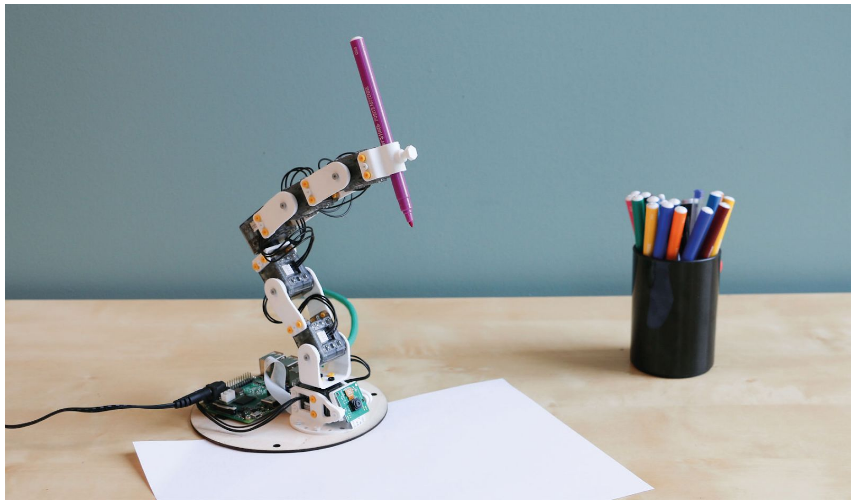
Erasmus+

EARLY

EARLY Teaching Scenario



Topic: Drawing with Poppy Ergo Jr

Aim(s): Students learn how to:

- Assemble Poppy Ergo Jr robot
- Use basic commands of Snap!
- Programme simple movements of Poppy

Skills pupils develop during the scenario: connect to curriculum →

The National Curriculum for Polish primary schools (IT) states that by the end of class 8, students should be able to:

- Design, create and test programs in the process of solving problems
- Use in these programs (i.a.): input / output instructions, arithmetic and logical expressions, conditional instructions, functions
- Design, create and test software controlling a robot or other object on the screen or in reality
- Search the network for information needed to perform a task, using complex forms of queries and advanced capabilities of search engines

The course components that are trained in this learning scenario are as follows:

- Creating a robot from provided components
- Programming the robot with a blocks-based graphical programming language
- Problem solving
- Collaboration

Target group: pupils in primary school (grades 7 -- 8)
Age of students: 13 - 14 years old
Number of pupils: maximum of 10 in a subgroup
Duration (estimated time/number of lessons): 3 sessions x 45 - 90 minutes each

Prerequisites (necessary materials and online resources):

- Poppy Ergo Jr with pencil holder
- Computers with sufficient parameters to run Snap!
- Online support materials available at https://www.poppy-project.org/en/robots/poppy-ergo-jr

Introduction to the scenario *(incl. possible applications, alternatives and risks)*

Poppy Ergo Jr is a robotic arm designed for education. It consists of six motors allowing a range of movements and 3D printed elements. It comes with three tools for different interactions with its environment: a gripper, a lampshade and pen holder. The robot is controlled with the Raspberry Pi board and a camera makes it interact with the objects around. Its movements can be programmed with a visual programming tool that gives students a variety of creative and playful learning experiences. Although the producers claim that the robot is easy to build/modify and everything in it is transparent and accessible to promote an educative approach we have discovered that Poppy Ergo Jr actually poses a number of challenges for a successful workshop experience. The following scenario proposes a pathway to follow if you want to avoid some of the pitfalls in exploiting the evident potential of the robotic arm in primary school education.

Before the program begins (preparatory work for teacher)

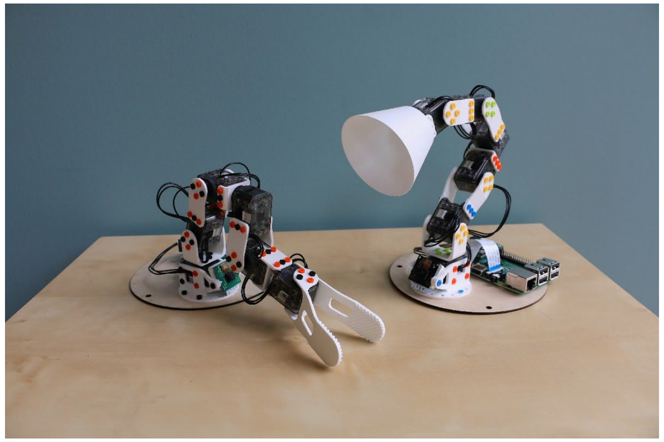
- Prepare a computer lab with enough work stations for the students.
- Poppy relies on Snap! (https://snap.berkeley.edu/), a variant of Scratch. Ensure that all the computers have a stable Internet connection and can run Snap (open a browser window and connect to http://snap.berkeley.edu/run).
- The number of robotic arms available will impact the size of groups to be involved in the workshops. The components are numerous and some tiny, so not more than 2 - 3 students should work on one robot's assembly. Most of the parts can be 3D printed to reduce the cost of building the robots - print the sufficient number of these components and purchase only the required electronic and mechanical devices.
- First go through the whole process of constructing and programming the robot yourself - we advise you to reserve at least one weekend for this playful and engaging experience before you challenge the students with the task!

Main part of the scenario (3 lessons)

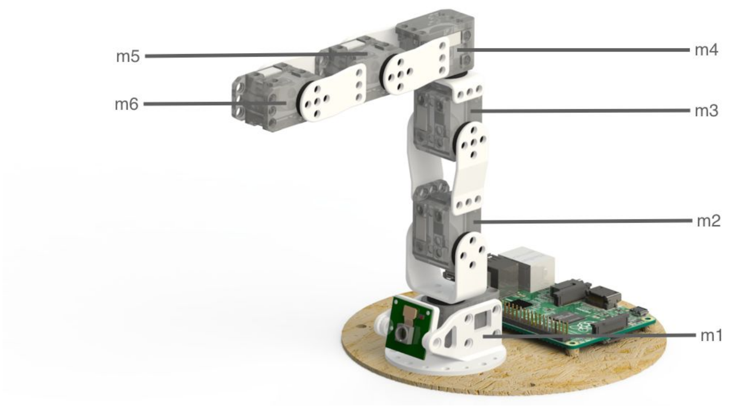
Each of the following lessons is a distinct unit of learning that can be delivered on its own. Ideally, all the students progress through all the steps but this depends on their level and the number of computers and robots available for the workshops. Perhaps the best idea is to run the first lesson devoted to assembling the robot with a small group of dedicated students constructing 1 - 3 robotic arms and then the following lessons with the whole class.

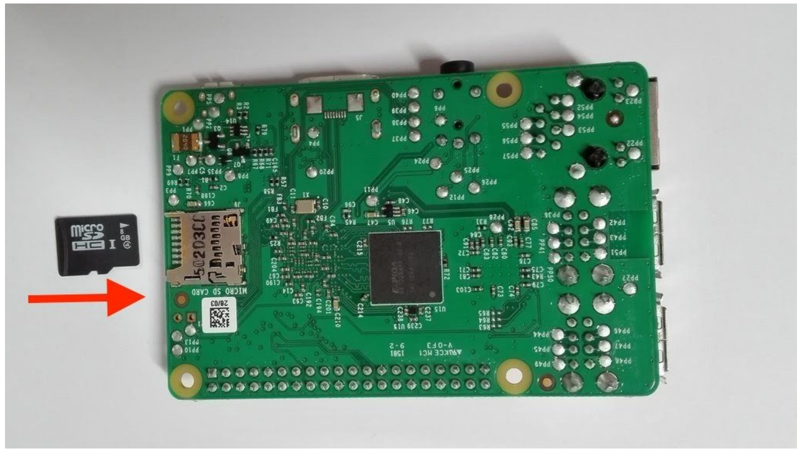
Lesson one: Building Poppy Ergo Jr

The producers published a brief guide on how to assemble the robot which covers motors configuration, electronic assembly and hardware construction https://docs.poppy-project.org/en/assembly-guides/ergo-jr/index.html This resource should be the main frame of reference during the whole process although it is certainly not for the students to read it in its entirety. We developed a video tutorial for beginners to facilitate the construction process https://edurobots.eu/assembling-poppy-ergo-jr/ Start with watching this tutorial from beginning to end. Then watch it again and start assembling. Stop and ask the teacher if you are not sure about the next step.



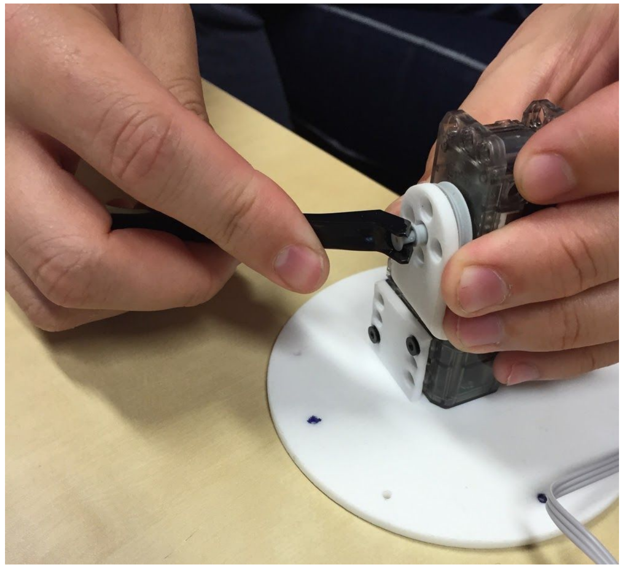
Motors configuration: in order for the motors to be connected and identified on the same line, they must have a unique ID. You will get them all set to the same ID. You should set a new and unique ID to each of the motors.





Electronic assembly: insert the micro-SD card inside the Raspberry Pi - push the micro-SD in the connector slot until you hear a "click" sound.

Mechanical assembly: it is done with special rivets that you will need for connecting different parts of the robot. The orientation in which you assemble them is important so make sure that you follow these guidelines closely.



It is a good idea to start with letting the pupils try out the ready-made robot as a motivation-enhancing factor. Then they will be really keen on and have the patience which is required to assemble their own.

Before you finish the lesson let the students test their robots. Go to your robot home page http://poppy.local. Click on *Reboot the robot* button to be sure that the robot software was started with all motors wired in. You can go to the monitor, click on *Monitor and Control* button. If the robot software is correctly started, you should see the green connection logo, otherwise it will be red. If the connection logo is red, you can see what is wrong by looking at messages in *What happened?* page. Most of the time, it's because a wire is unplugged or because you forgot to configure a motor.

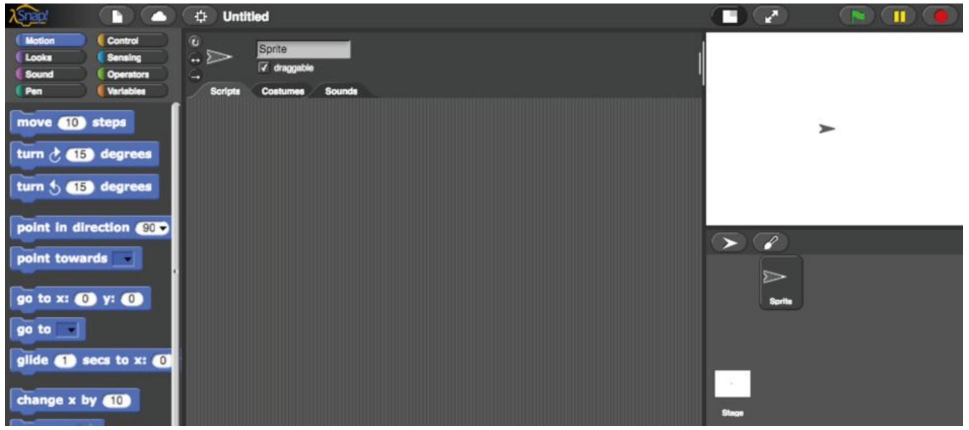
Lesson two: Connecting Poppy Ergo Jr to Snap!

Snap*!* is a blocks-based graphical programming language that allows users to create interactive animations, games, and more, while learning about mathematical and computational ideas. Snap*!* is open-source and it is entirely written in javascript, you can use it from the official website but you can also use a copy of the website in your personal computer and open the snap.html file in your browser. The students will need to know the basic functions of Snap! to be able to programme the Poppy robot. This lesson has this purpose.

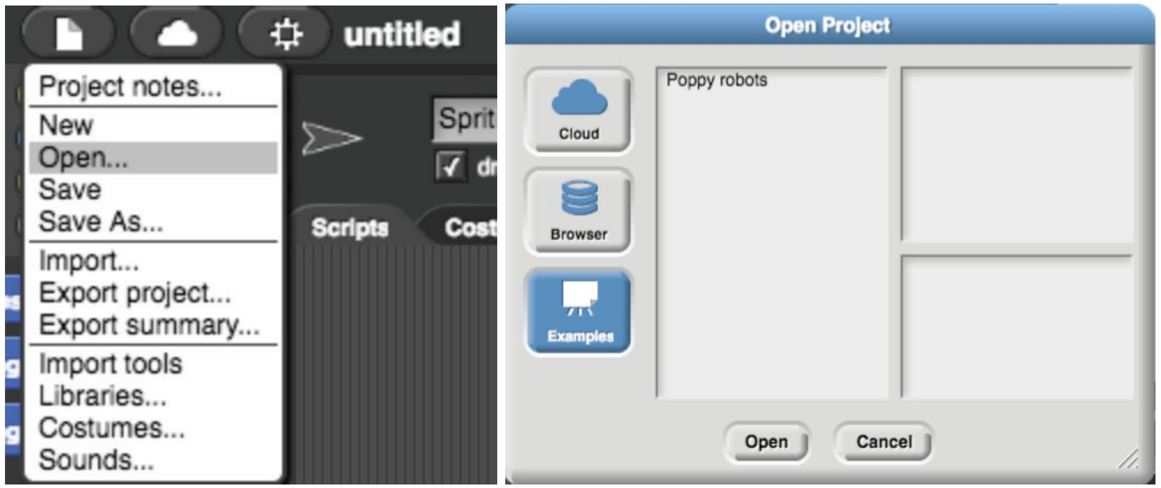
First students have to choose Snap! from the main menu.

The Snap! is divided into three parts: To program Poppy Ergo Jr, you will mainly use the zones left and center.

- Left: the list of available blocks (instructions), arranged in different categories.
- In the center: the script area, where we assemble the blocks.
- Right: above a display area that can be controlled by scripts, below programmable objects that can evolve in the display area.



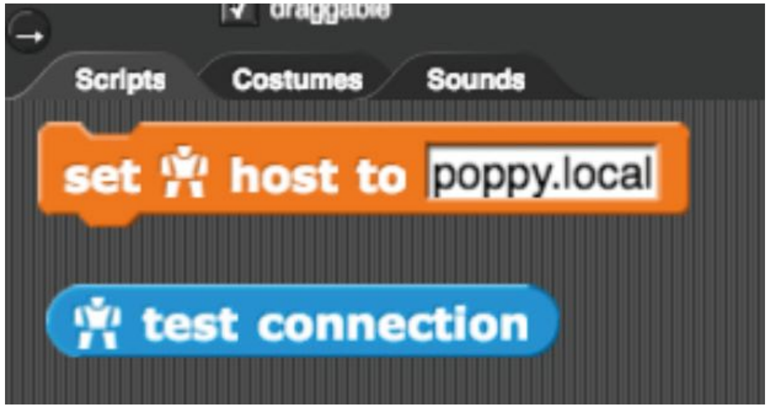
Open the Poppy robots project to be able to access the specific blocks for the Poppy robot: to do this click on: File> Open> Examples> Poppy robots> Open



Tip: For information on a Snap! right-click on the block then select help.

You should see two blocks appear in the script area.
Check that you are connected to the robot:

- Write the name of the robot or the IP address in the block (Here the name of the robot is poppy)
- Click on the block TEST CONNECTION to verify that you are well connected to the robot:
- If an error message appears, check that you are connected correctly (see the robot documentation, if necessary).



Save your program: you can save it on your computer's hard drive by clicking on: Files> Export> right click> Save as or save it to the cloud, which requires having a user account:

- Click on the cloud menu in the toolbar
- Select the sign up option from the menu, and follow the instructions
- Check your email to get your initial password

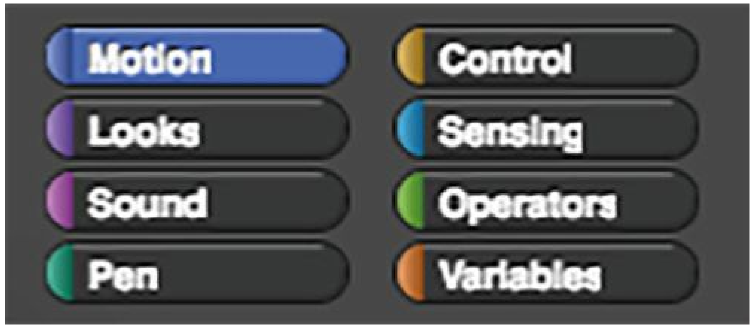
Before activating Ergo Jr, check the correct position of the robot each time. You are ready to start the activities!

Lesson three: How to draw with Poppy Ergo Jr

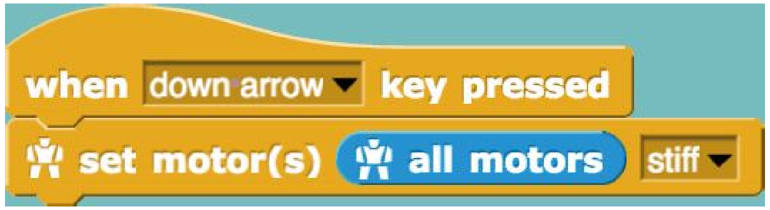
In this lesson the students will learn how to make the robot draw using the Snap! - a language of programming with which we assemble instruction blocks. An assembly of blocks is called a script.

To find blocks in Snap ! you can search for:

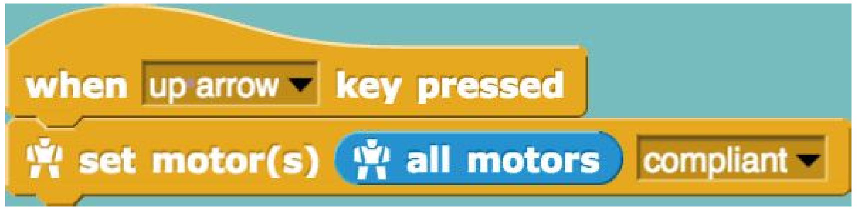
- By color / category (each one color category)
- By keywords (> Right click on the left side> find blocks) and enter the keyword robot in order to select only the specific blocks for Ergo Jr robot



Create the two scripts below in order to be able to put Ergo Jr in specific positions:



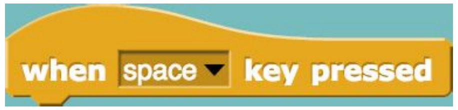
```
when [down arrow] key pressed
set motor(s) (all motors) [stiff]
```

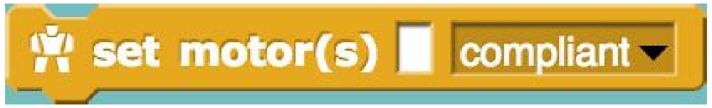


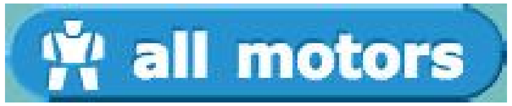
```
when [up arrow] key pressed
set motor(s) (all motors) [compliant]
```

In order to do this:

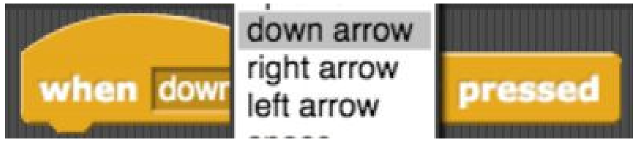
1. Select and drop the three blocks which you need on the central workspace:

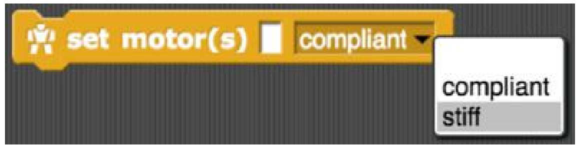




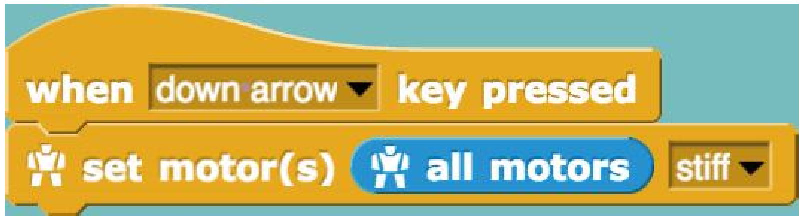


2. Click on the lists and choose the proper values:



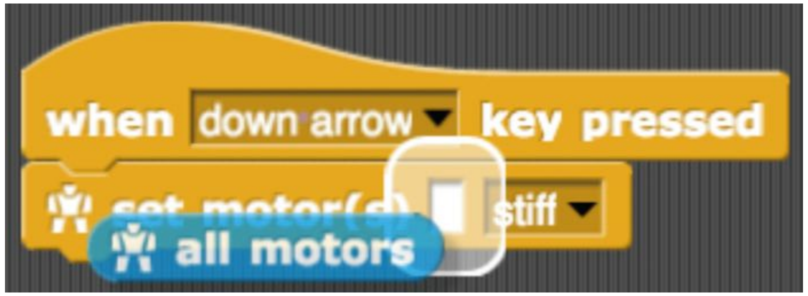


3. Assemble the blocks together:



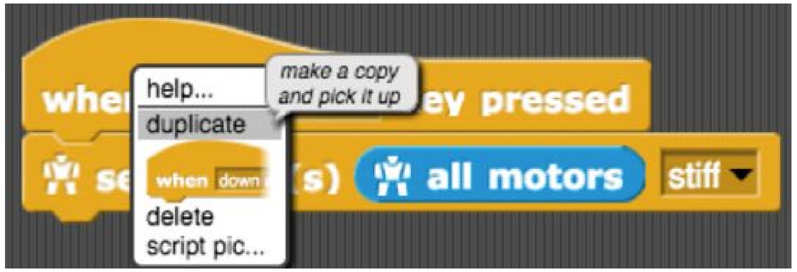
4. Do the same for the second script.

To nest blocks: select
the block and drop it in the desired location



with the mouse. The white border
indicates that the blocks are going to fit together.

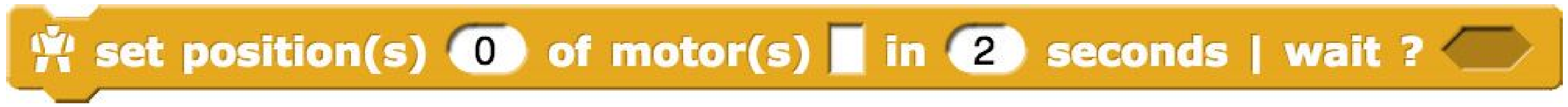
You can copy / paste the blocks
and scripts: Right click then duplicate



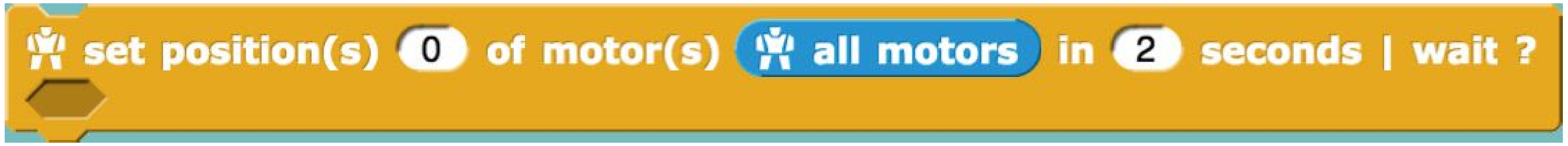
Activate both scripts (a white border appears around a script activated): Press ⇩ or ⇧ on your keyboard and manipulate the robot to put it in different positions.

Get Ergo Jr moving with his motors

We use the following block to move Ergo Jr, motor by motor:

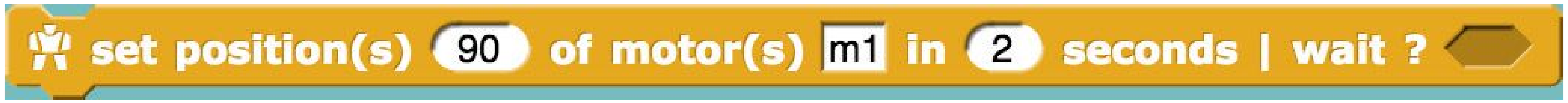


Before you can get moving with Snap! make sure that all the motors are activated (stiff mode) and put all the motors in the basic position (which corresponds to the position where each motor is at 0 degrees: aligned with the notch) in clicking on the following block to execute it:



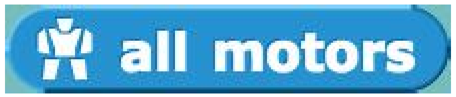
```
set position(s) 0 of motor(s) all motors in 2 seconds | wait ?
```

Put the motor m1 in the 90 degree position in 2 seconds.

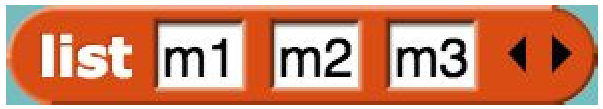


```
set position(s) 90 of motor(s) m1 in 2 seconds | wait ?
```

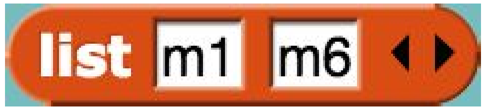
Look for the blocks opposite and execute them in turn:



```
all motors
```

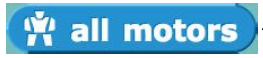
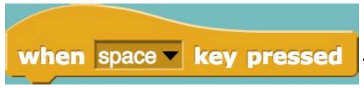


```
list m1 m2 m3
```



```
list m1 m6
```

The blocks have different shapes, each shape corresponds to a specific category:

- The oval shaped blocks (like  all motors) are called reporters: when executed, they return a value.
- At the top of the script can be a Hat block, which indicates when the scenario should be executed. The names of Hat blocks usually start with the word When (example:  when space key pressed); a script does not necessarily have a block Hat, but without this block, the script will be executed only if the user clicks on the script itself.

- Command blocks (like

```
set motor(s) [ ] compliant ▾
```

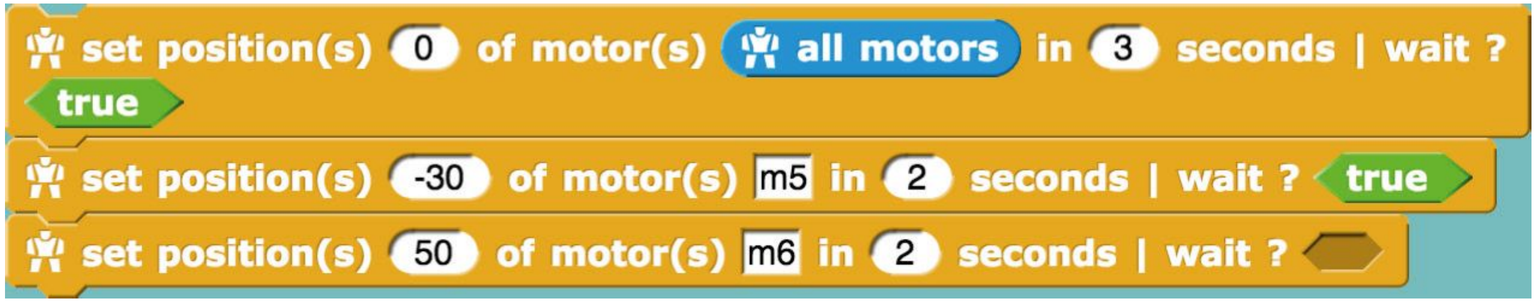
) correspond to one action.

Modify the set position block to set the motor m1 and the motor m6 in the -30 degree position in 2 seconds.

With the help of the blocks we have just discovered, build two programs corresponding to the instructions below:
- Press ⇨ on your keyboard, then put all the motors in 0 degree position in 3 seconds.
- Press ⇦ on your keyboard, then put the m1 motors and m4 in position 60 degrees in 2 seconds.

Now let's use the motors to create movements.

Run the script below and watch what happens:



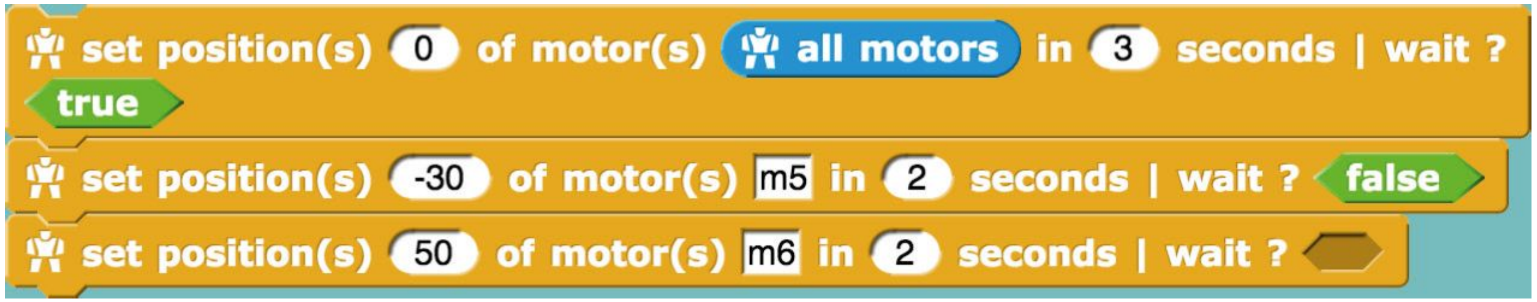
```
set position(s) (0) of motor(s) (all motors) in (3) seconds | wait ? <true>
set position(s) (-30) of motor(s) [m5] in (2) seconds | wait ? <true>
set position(s) (50) of motor(s) [m6] in (2) seconds | wait ? < >
```

Replace the second block

```
<true>
```

with the block

```
<false>
```

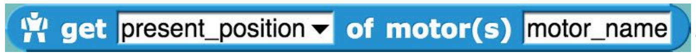


```
set position(s) 0 of motor(s) all motors in 3 seconds | wait ? true
set position(s) -30 of motor(s) m5 in 2 seconds | wait ? false
set position(s) 50 of motor(s) m6 in 2 seconds | wait ?
```

When there are two (or more) nested blocks, in what order do they work the actions ? What happens when WAIT is set to true? What happens when WAIT is equal to false?

The lines of code execute almost instantaneously; and sometimes even if the position requested in the previous line has not been reached. The WAIT part allows you to wait until the engine has reached the desired position before executing the following command.

With the blocks you know now, play a movement to Ergo Jr meaning “hello” when you press the b key.

- Start with a simple movement then enrich it as you go.
- Choose the engines you want to use for the creation of the movement.
- Make the robot play the chosen movement (in compliant mode) and observe the actions of each engine.
- You can help yourself with the block  `get present_position of motor(s) motor_name` to know the position of a target engine, and thus note the value to use it again.
- Program the movement motor by motor and test each time the result of your program.

Do not hesitate to create and test other movements!

Learning outcomes
Students will be able to:

- Work independently, trusting themselves
- Follow written instructions and video tutorials
- Assemble a small robotic arm with provided components
- Use a block-based programming language to make the robot execute movements
- Collaborate with peers in on the workshop tasks

Sources
This scenario is based on the materials available at https://www.poppy-project.org/en/robots/poppy-ergo-jr:
- Apprendre à programmer Ergo Jr en Snap! Livret d'accompagnement du robot Poppy Ergo Jr (Conception et réalisation : Équipe Flowers Inria et Ensta ParisTech et Poppy Project; Design graphique : Antonin+Margaux)
- A day to build and program your robot! (https://www.poppy-project.org/en/posts/poppy-ergo-jr-workshop-at-cern)