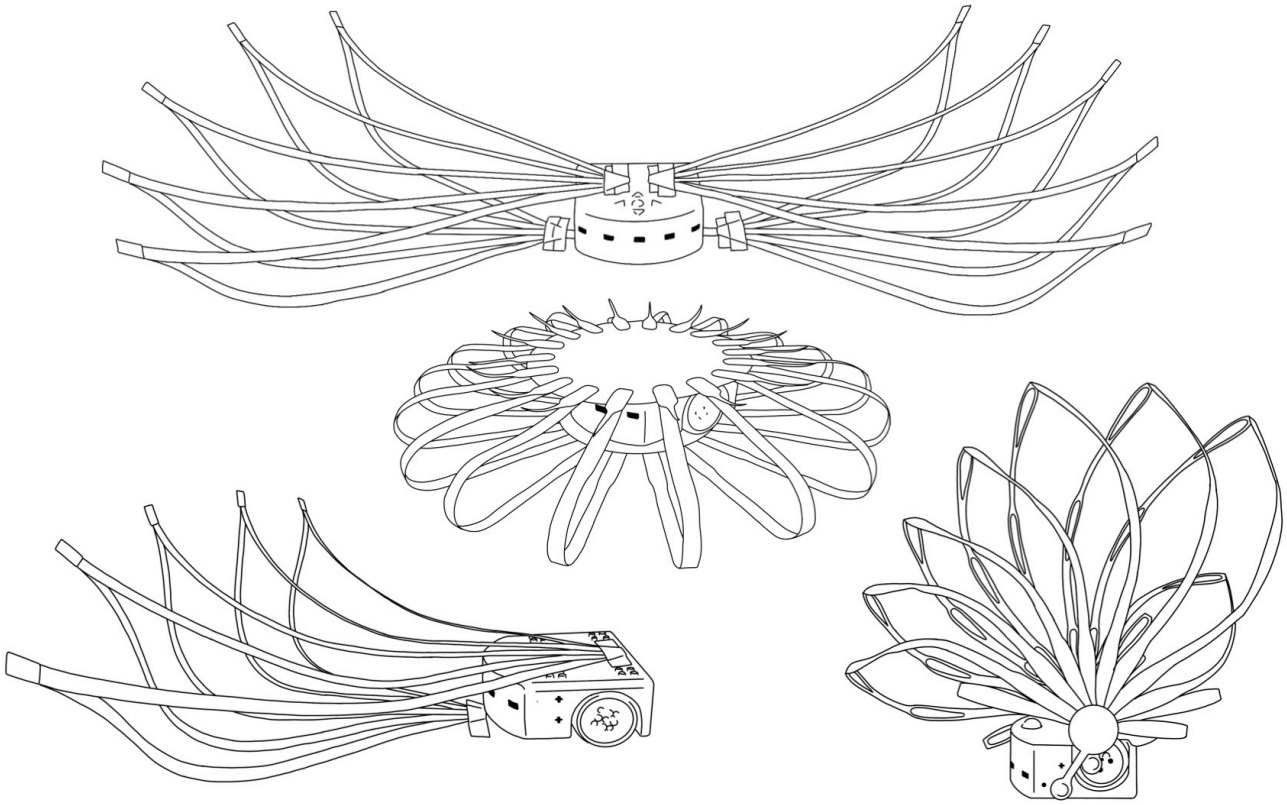
Semester Project

MOBOTS group

Designing Costume Behaviors



Student :

Emilie Grandjean

(286734)

Supervisors :

Léa Pereyre

Manuel Bernal Lecina

Professor :

Francesco Mondada

Lausanne, Spring 2024

École Polytechnique Fédérale de Lausanne

Designing costumes behavior

Professor :	Francesco Mondada	Office	francesco.mondada@epfl.ch
Scientific Assistant Contacts :	Léa Pereyre	Office	lea.pereyre@epfl.ch
	Manuel Bernal Lecina		manuel.bernallecina@epfl.ch

Project Type :	Semester Project	Section :	Microengineering
Official Start Date :	20.02.2024		
Submission of Final Report :	03.06.2024		
Final Presentation :	11.06.2024		

Designer Léa Pereyre has developed a set of costumes around the Thymio robot, which the robot animates with a few simple movements (constant wheel speed, linear back-and-forth movements, in some cases avoiding or following lines). Despite the very interesting aesthetic results obtained with this approach, some preliminary explorations have shown that better use of the Thymio's sensors and more sophisticated programming could considerably enrich the aesthetic effects.

The project goal is to explore in a systematic way :

- All the Thymio's sensors to define which can be used to interact with existing costume types, giving a few examples of how they can interact with a few demonstrations
- Which interaction possibilities could be used to interact with an external observer to make the system interactive, giving as well some examples of how they can interact with some demonstrations
- Which programming languages (VPL3, Blockly, Python, ...) are interesting/necessary for programming this or that interaction, and how far they allow you to go.

The approach of the project has to stay compatible with the existing goals, that are being compatible with an educational context and being focused on aesthetic results.

Contents

1 Introduction

My project is based on the work of Léa Pereyre in the MOBOTS group. Léa Pereyre has developed a set of costumes around the Thymio robot ("Thymio") which the robot animates with a few simple movements (constant wheel speed, linear back-and-forth movements, in some cases avoiding or following lines).

Despite a very interesting aesthetic results obtained with this approach, the idea of my project is to explore a better use of the Thymio's sensors and a more sophisticated programming approach while keeping in mind the original, aesthetic and educational aspect of the use of the Thymio.

The main goals are to implement a more advanced programming approach, using various programming languages such as Python, VPL3 and Blockly, to exploit the full use of the Thymio sensors in combination with the existing costumes and finally to explore ideas on how to make my research compatible in education purposes. This is to push this project in a programming aspect and not only in a creative, aesthetic aspect.

Hence, I divided my project in three major parts : an analysis of the sensors and costumes pairing for an estimation process, a coding aspect starting with the Python language then adapted to VPL3 and Blockly, and at last, creating educational activities based on the programs in these various languages.

2 Methodology

As a first step, I needed to determine all the different sensors of the Thymio and which ones could be used as part of this project. There is a total of 19 sensors that can be classified in 5 different groups : the proximity sensors (9 in total), the buttons (5 in total), a temperature sensor, a microphone and accelerometer sensors (3 in total). I decided to use all of them except the temperature sensor which did not make much sens within the objectives.

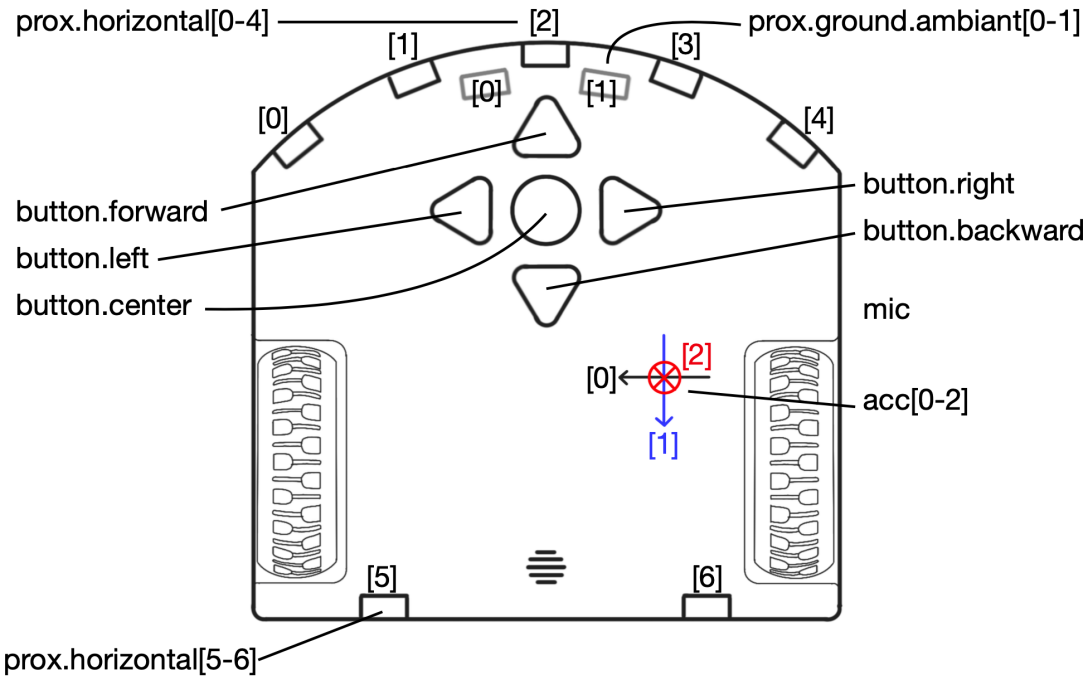


Figure 1: Schema of the Thymio Sensors

As we can see in the figure 1 above, these are the placement of the different sensors used as part of this project. This graphic is based on Reference [1], a Thymio Cheat Sheet.

The second step was to draft a table of sensors and costumes pairing and proceed to an estimation process. This allowed me to sort the sensors and list them depending on their possible use with the existing costumes (Appendix 1 - 3 found in the Appendices section).

By analysing previous videos made by Léa Pereyre, the resulted table, seen in the figure 2 below, is the estimation of the sensors use depending on the costumes. The cross-section (analysis) between the costumes and the sensors is color coded to analyse the possibilities and restraints for each case.

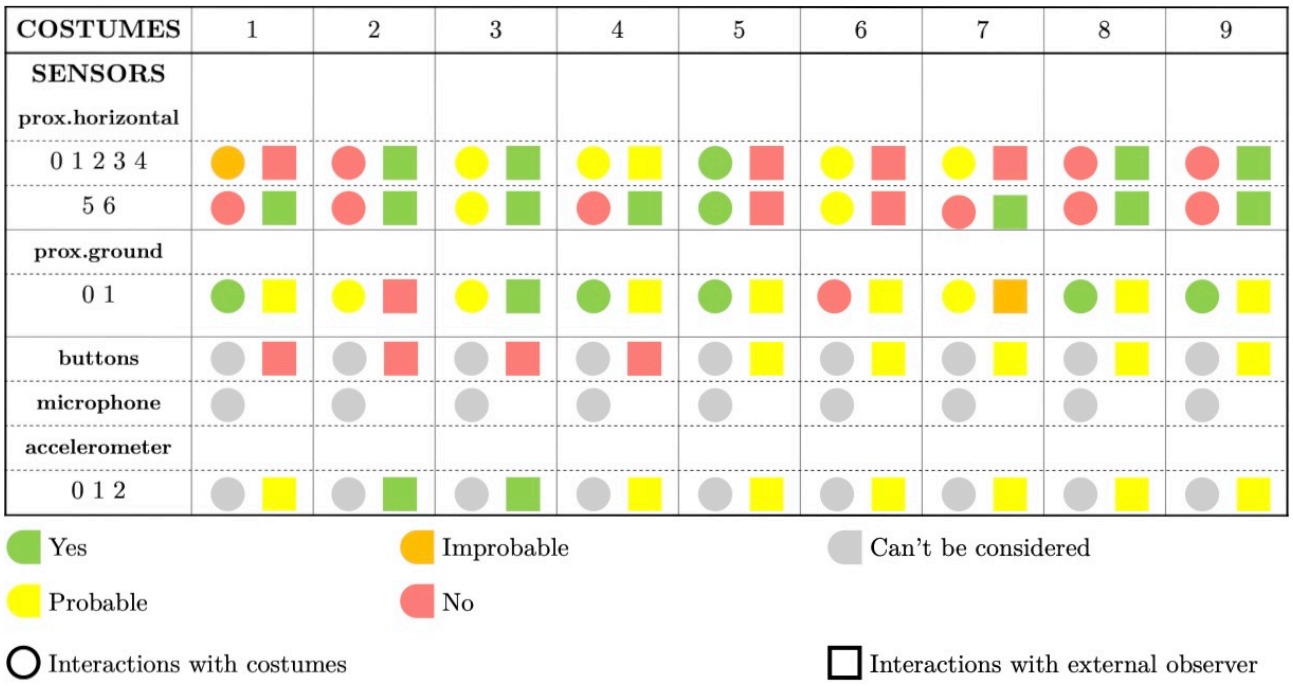


Figure 2: Table of sensors & costumes

From this first table, figure 2, some similarities in the possibilities of the use the sensors between several costumes can be noticed. This led to the creation of the second table, figure 3, where the costumes were classified in 3 subgroups. It reduced the number of costumes to work with, allowing a less redundant analysis and experiments/ testing.

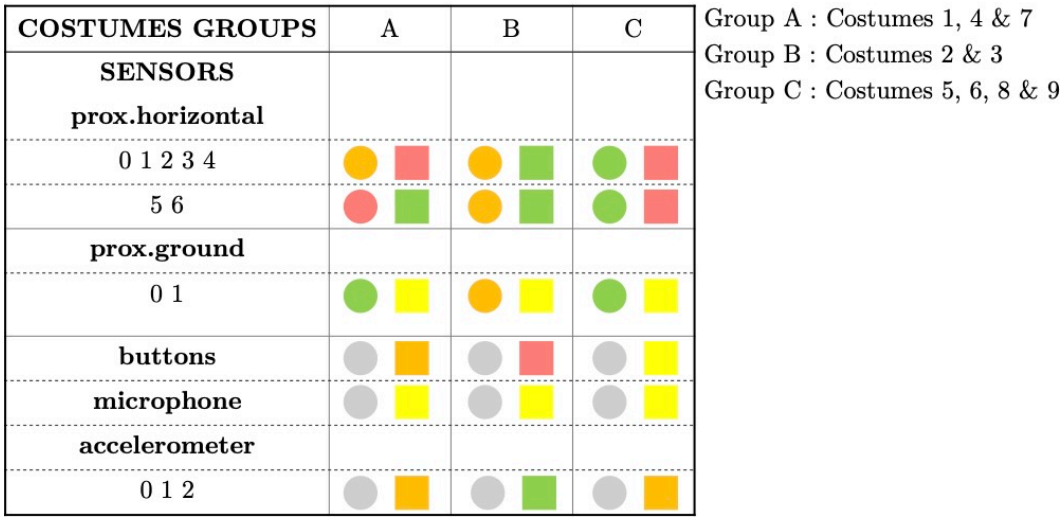


Figure 3: Table of subgroups of sensors & costumes

These tables and their analysis allowed me to develop a work strategy for the next steps. The decision made was to start the experimentation and programming for the first subgroup of costumes using the Costume 1. The idea was to develop a base of the more advanced programming approach in Python, figuring out some solutions, possible costume modifications and ameliorations that could be applied to the other costume subgroups.

The details of the Python programming approach is discussed in the subsection 3.1 Python of the Programming Approach section 3.

The process was to study the use of each sensors one by one and developing either an interactive system or an autonomous system and repeating this iteratively with each costume groups.

Once the Python programming completed, the next step was to implement the same programs to the other languages : VPL3 and Blockly. This allowed to see which sensors, costumes or program are better suited for which programming language. The idea is to compare their usefulness and their level of difficulty if adapted to the educational framework.

Then, after testing and comparing the different programming languages, the final goal is to adapt all my research and analysis to the educational context. For this aspect, the objective was to create an exercise sheet per programing language based on a course handbook published by the DGEO (*Direction générale de l'enseignement obligatoire et de la pédagogie spécialisée*) (Reference [2]).

3 Programming Approach

The programming approach was first focused on the Python language with the Costume 1 before moving to the other costumes and then the other programming languages. The idea was to adapt the same Python code in VPL3 and Blockly to be able to compare their use and which would be more optimal for the different costumes and programs.

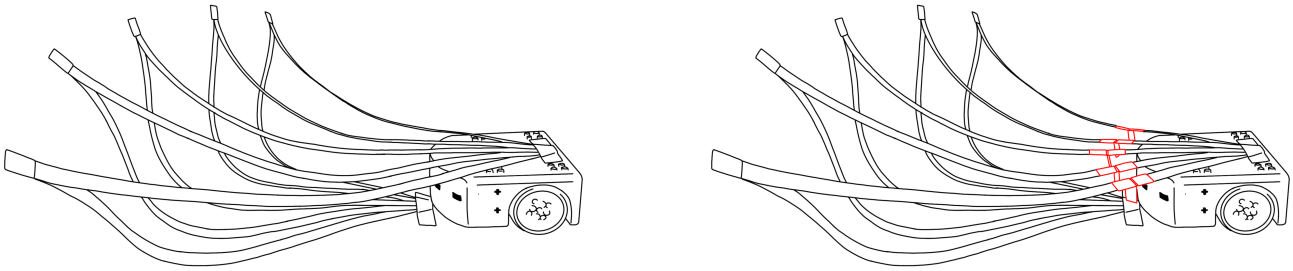
There was a huge part of creative aspect because the program coded for the Thymio and the different costumes was entirely up to me. I could implement whatever I wanted. However, I restrained myself to consider the Thymio and the costume as one rather than separately.

3.1 Python

For the Python programming approach, I started by working on the Original Costume 1. To build the program, I began with coding the proximity sensors in order to build an interactive system with an external observer.

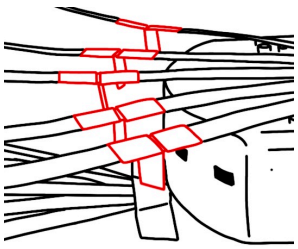
Some issues came up because the front proximity sensors were not accessible to an external observer and because the costume could not trigger them. After careful consideration, a couple ideas came to mind.

At first, if I wanted to have an automatic system where the costume would trigger the sensors, the costume had to be modified. So, I proceeded to modify the costume in order to adapt it to the desired outcome.



(a) Original Costume 1 (b) Modified Costume 1

Figure 4: Modification of the Costume



The modification that was made was the addition of little extrusion of paper fitted on the costume. These added props allowed the front proximity sensors to be trigger by the costume and could induce some movements until they got out of the sensors range.

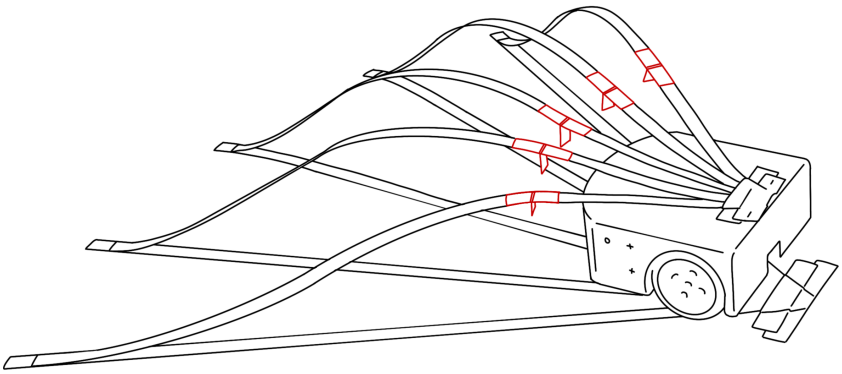


Figure 5: Modified Costume 1 when proximity sensors aren't triggered anymore

This prompted me to think about how this costume could be further modified enabling an access to all the proximity sensors for an interactive system. This led to the second modification of the Original Costume 1 by doubling it and placing it on either sides of the Thymio rather than in the front, as seen on the figure 6. This provided full access to all horizontal proximity sensors and offers more opportunities for interaction with an external observer.

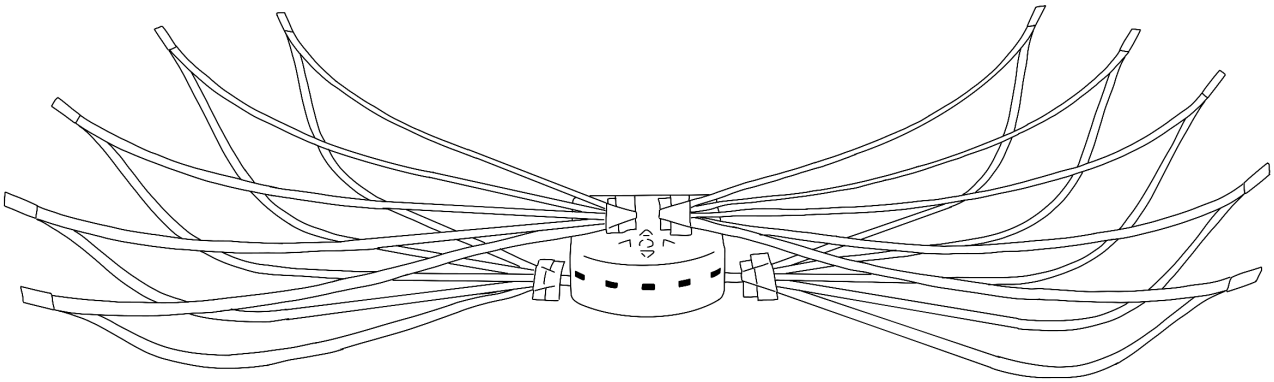


Figure 6: Modified doubled Costume 1

This proposition allowed extra applications with the costume without removing the original aspect and movements of the Original Costume 1. Since this concept was really appreciated by Léa Pereyre, a decision was made to create a new costume based on this adaptation, as can be seen on the figure 7 ("Doubled Costume").

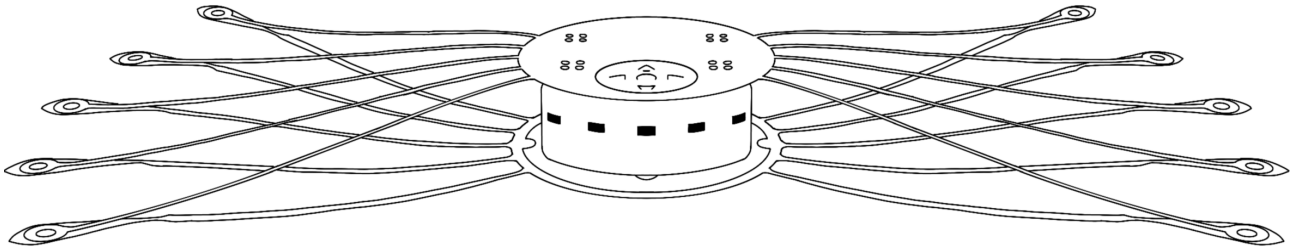
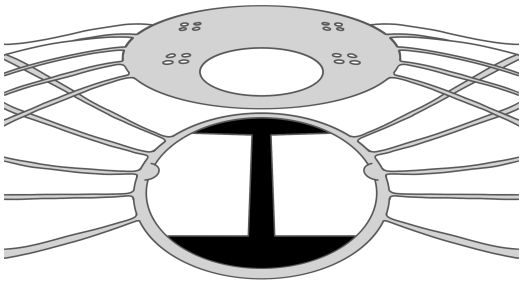


Figure 7: Doubled Costume

Another interesting aspect with these costumes (from costume group A) is that the ground sensors are usable as well. In the Python code, at first, I just implemented a function which would trigger the Thymio's movements as soon as it crosses a black line, for instance.



The Doubled Costume incorporates a dedicated slot in which a black line can be placed and crossed by the Thymio, triggering a desired action.

Once all the proximity sensors (horizontal and ground) were implemented, I continued the same train of thoughts with the next sensors and realized that in the case of the Costume 3, the interesting sensors to use would be the accelerometers. The idea is to have the costume movements depending on the inclination of the Thymio while held by an external observer.

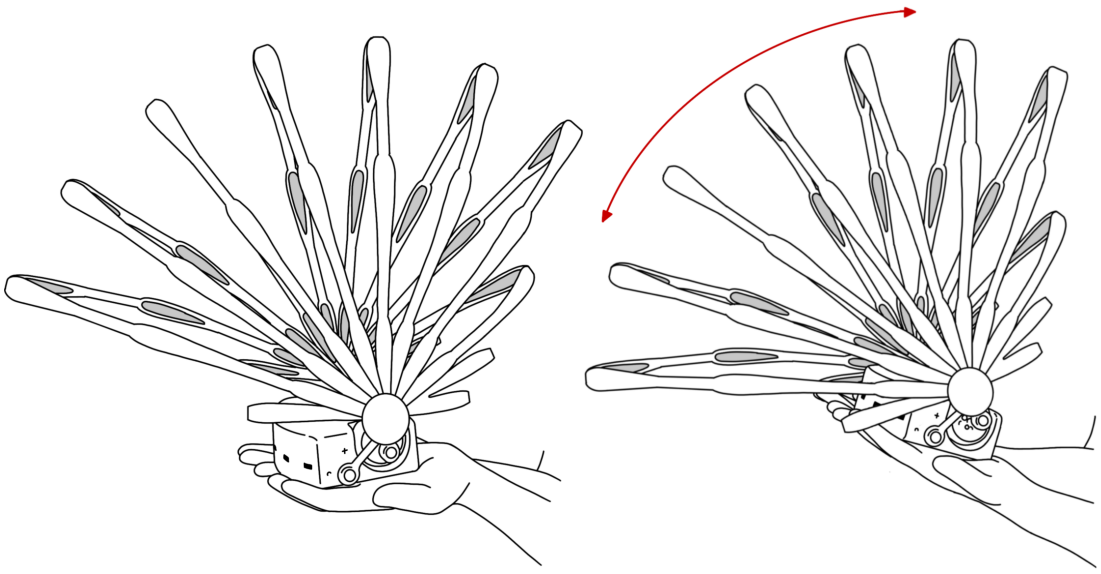


Figure 8: Costume 3 held in hand

To implement the accelerometer sensors, I started with a testing process. I did this by printing the values of the various accelerometer sensors (3 in total) in order to understand and draft a list of which sensor was related to which movement and what were the values they would take in the different inclination positions.

Accelerometer sensors	Front	Back	Left	Right
0	no effect	no effect	max 22 to 25	max -18 to -20
1	max -21 to -22	max 20 to 22	no effect	no effect
2	max 0 to 1	max -2 to 0	max 0 to 1	max -1 to 1

After this analysis I was able to create my code depending on the observations I made. The idea was to divide the movement into five separate conditions, whether the Thymio was tilted on its left, right, forward, backward or just horizontal.

As for the microphone sensor, the idea was to use it as part of a possible class activity. I was able to create a function depending on the sound heard by the Thymio (a clapping noise for instance), but I quickly realized that this idea was incompatible with a classroom full of students. The optimal environment with as little surrounding noise as possible is necessary to avoid an unintended triggering of the Thymio's microphone. The conclusion regarding this sensor was therefore to leave it out of my project.

With respect to the button sensors, my goal was to implement different programs linked to them. The idea was that the center button would be used as a **stop** button, meaning it would exit the current program. The arrow buttons would be to select various customized programs.

However, when I configured a button and the Thymio entered the specific program, it stayed stuck and was hard to get out of (I had to press on the **stop** button at the exact right time to exit the program). I needed to find a solution to this problem which brought the idea to implement ***Threads***.

The concept is to have three ***Threads***, one being the *Event List* (*to classify the sensors in order of priority*), another a *State Machine* (*to link the buttons to the different programs*) and the last one the *Actions* (*to implement the different Thymio movements depending on the selected program*).

The *Event List* assigns the highest priority to the center button, followed by the other buttons, the proximity sensors, then the accelerometer and finally the microphone.

After working on the ***Threads***, I realized that the *State Machine* and the *Actions* ***Threads*** were not necessary. Instead, I created 5 little ***Threads*** each corresponding to a specific button into which I would just need to call the functions from my specific python file in order to implement the wanted actions. This allowed me to start a button ***Thread*** when its button is pressed and kill the button ***Threads*** when the **stop** (center) button is pressed.

In this situation, one can enter a specific customized program by pressing on an arrow button but cannot access the other arrow button programs until the **stop** (center) button is pressed, which puts an end to the customized program.

The arrows allow to access the different customized programs, three of which correspond to the different costume groups and the last arrow button is for a random movement program, i.e. it will go into an automatic system for the Costume 1, for instance.

The interest in dividing the code for the different costume groups is due to the fact that some sensors are more suitable to be used for some costumes and less for others. This was analysed and tested out based on the Table of subgroups of sensors & costumes seen in the Methodology section 2.

Event List

0. Center button 1. Other buttons 2. Proximity sensors 3. Accelerometer 4. Microphone

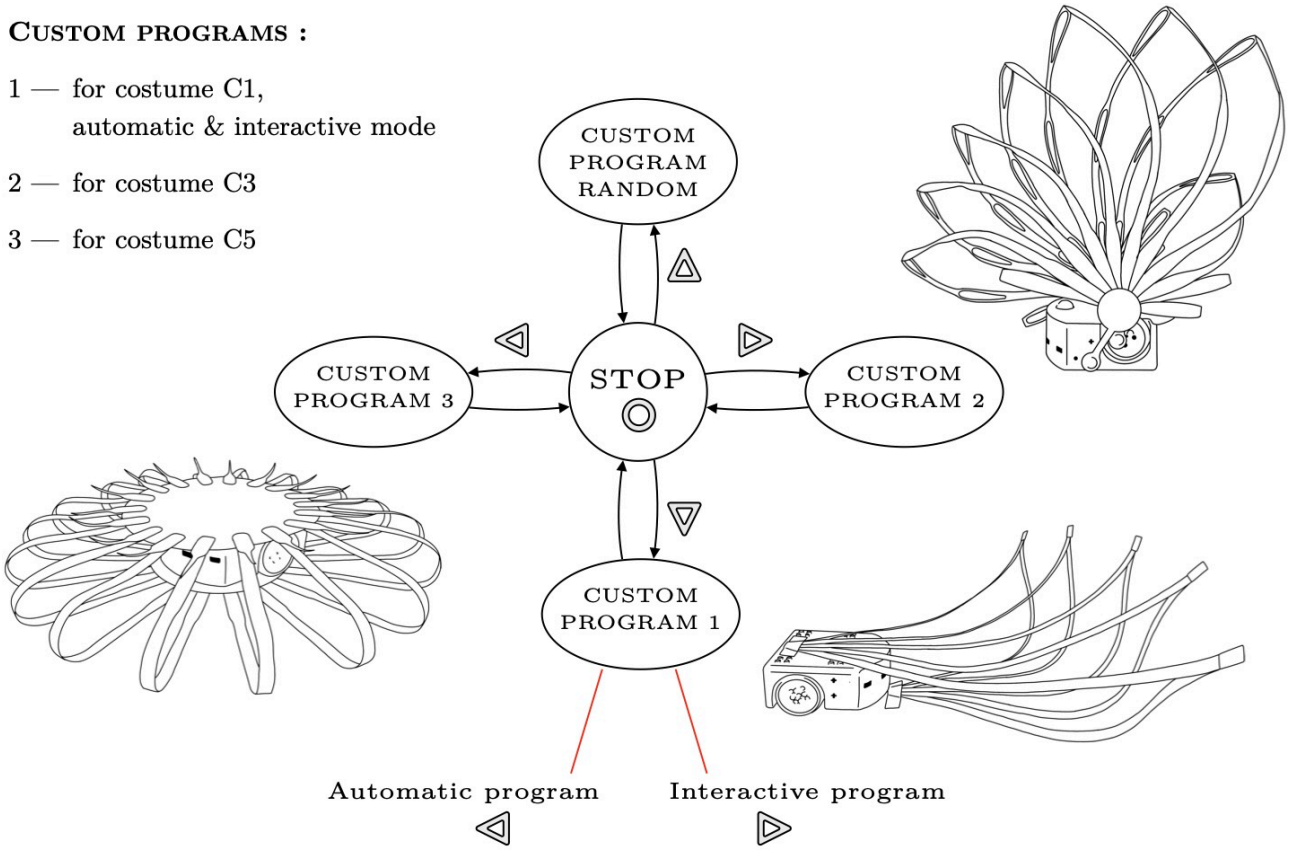
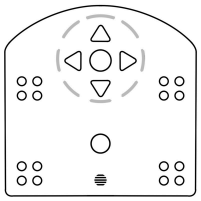
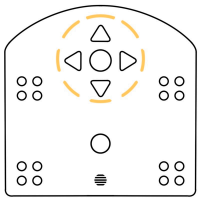


Figure 9: State Machine - Buttons list threads

Button Center *Thread*

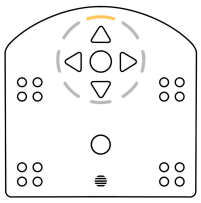


No program : When one runs the Python program, the Thymio is in this state, no light and no button ***Threads*** turning and waits until a button is pressed. The *Event List* ***Thread*** classifies the priorities and verifies the states of the sensors. This is also the Thymio's state when one presses the **stop** button.



Center Button program : The way the code is written allows to use the center button as a **stop** button but as well as a specific custom program. When one presses on the center button while being in the no program setting, all the circle LEDs light up and the Thymio's state will be corresponding to this program.

Button Forward *Thread*



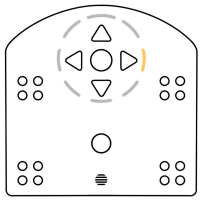
Forward button program : The custom program linked to the Forward button is a random program. Every time when entering this program the variable *random_integer* is set to a random number between 1 and 5. Various programs or actions can be assigned to each number.

Button Backward *Thread*

Backward button program : The custom program linked to the Backward button is implemented for the Original Costume 1 or its modified versions. When entering this program, the Thymio's movement is set to an automatic system where it moves from left to right in a loop.

It is possible to access an interaction system by pressing on the right arrow button while in this program. In order to return to the automatic system, one just need to press on the left arrow button. This allows the user to switch between either automatic or interactive programs for these costumes. The left and right LEDs light up depending on which system the Thymio finds itself.

Button Right *Thread*



Right button program : The custom program linked to the Right button is implemented for the Costume 3. This runs the function linked to the accelerometer sensors. This is an interactive system where the Thymio's movements are based on the inclination of the robot induced by an external observer.

Button Left *Thread*

Left button program : The custom program linked to the Left button is implemented for the Costume 5. This system induces the Thymio to rotate on itself until its proximity sensors are triggered and then switches the rotation direction until triggering the sensors once again. This produces an automatic loop movement.

3.2 VPL3

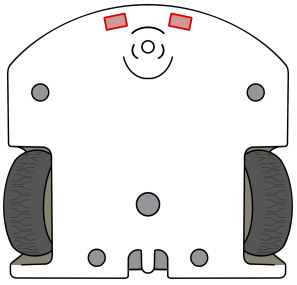
The goal to implement the VPL3 programming is to adapt the same Python code for the various sensors. As previously discussed, the Python code is separated in several functions adapted for the different sensors and costume groups. Therefore, I created different codes for each of these situations all recreating the same results as with the Python approach.

I started with the proximity sensors established for the Costume 1. This program uses the *horizontal proximity sensors* and allows an interactive system with an external observer. The implemented VPL3 code for proximity sensors can be seen in the Appendices section.

This application brought the next step of using the *ground proximity sensors*. In VPL3, it allowed for a program for line following. The code was inspired by one found in the documentation **[2]**, which was implemented uniquely for forward movements.

So, the interesting aspect to add to this program was to create an autonomous back and forth system, as can be seen in the VPL3 code for ground sensors.

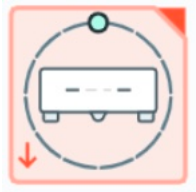
However, after some further testing, I realized that the line following moving backwards was not as reliable as the one moving forwards. Even though with a simple straight line it seemed to work well, it is noticeable that when the Thymio moves backwards, it does not follow the line very precisely. This can be explained by the fact that the *ground proximity sensors* are placed in the front of the robot.



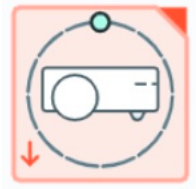
Moving forwards works fine because the *ground proximity sensors* placement allows to anticipate the movement as they detect the "future" position of the line being followed. On the other hand, when moving backwards, the Thymio cannot predict the "future" direction to take in order to follow the line since in this case the *ground proximity sensors* placement detect the "past" position of the line.

Depending on the shape of the line and the Thymio's orientation, the robot will possibly deviate in the wrong direction and hence, not follow the line anymore.

Some frustration came about with this realization because I was planning on creating the VPL3 activity on the principle to follow a line forward and backward allowing me to introduce the concept of FLAGS in programming. I had to rethink the exercise that could be used for this subject and programming language.



Then I adapted the code for the Costume 3 using the accelerometer. This was a task a bit more complicated to accomplish. The issue was that I started to implement the effects of the roll and pitch angles independently but they also need to be considered combined.



If both angle effects were only treated independently and if the Thymio was tilted towards the front and on one side, it appeared that the program could not decide which action had to be executed. Therefore, I needed to implement actions for both angles triggered.

This complication can show that VPL3 might not be the best programming language for the use of the accelerometer sensors, especially compared to the Python language. This is illustrated that the values of the tilting are pretty vague in VPL3. Conversely, in Python, the actions related to the degree of the Thymio's tilt can be controlled with high precision.

Then, for the Costume 5, the idea was to implement the autonomous program where the Thymio turns on itself until all proximity sensors are triggered, and then proceeds to turn in the opposite direction and so on. The VPL3 code for Costume 5, was the simplest to implement consisting of only 3 lines of code.

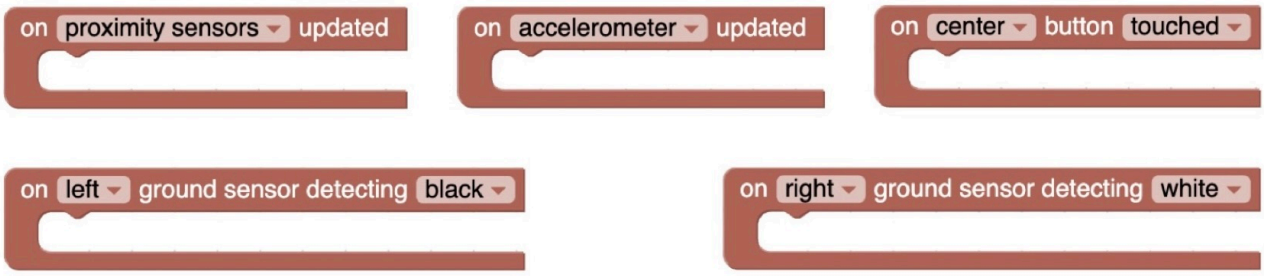
The only necessary aspect to keep in mind was to implement a type of FLAGS allowing the Thymio to know in which direction to rotate. In this case, I used the LED color to distinguish the turning direction. When the Thymio lights up in green, it turns clockwise until triggering the *horizontal proximity sensors*. At that moment, the Thymio's color is set to pink which indicates to the robot that it now needs to rotate counterclockwise until triggering the sensors once again, and so on.

3.3 Blockly

Then, after implementing the VPL3 codes, the final programming step was to adapt the same programs in the Blockly language. For this, I proceeded again in the same manner as with the Python and VPL3 languages, first the proximity sensors, then the ground sensors, the accelerometer sensors and finally the autonomous rotating program for the Costume 5.

It was actually quite simple to adapt the Blockly code for proximity sensors. However, when it comes to the Blockly code for proximity sensors and the Blockly code for forward line following, it is important to bear in mind the order of the code lines. This is especially true for the ground sensors : the condition to stop the motors when reaching a black line has to be placed first, otherwise it will not be taken into account and the Thymio will not stop at the line.

To implement the code in Blockly, it was necessary to use the *"on *specific sensors* updated"* event blocs. This allows the code to run at each update of the *specific sensors* values. The *specific sensors* correspond to the senors currently used in the program, either the *horizontal proximity sensors*, the *ground proximity sensors* or the *accelerometer sensors*



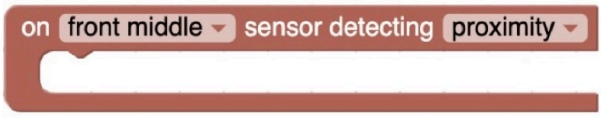
I first implemented the Blockly code for proximity sensors. It basically does the exact same thing as the interactive Python code for the *horizontal proximity sensors*. When the external observer only triggers the front middle *horizontal proximity sensor*, the Thymio backs up. When they trigger any of the front or back *horizontal proximity sensors*, the robot follows the users hand. When both of the back *horizontal proximity sensors* are triggered at the same time, the Thymio starts advancing until the sensors are no longer triggered.

As far as line following is concerned and based on the observation made with respect to the VPL3 section 3.2 above, I decided to implement the Blockly code for forward line following only.

For the Blockly code for accelerometer sensors, I used the same values previously studied in the Python section 3.1. Therefore, I do not think that this programming language is very interesting to use with the *accelerometer sensors*.

Finally, for the Blockly code for the automatic rotation program, the interesting aspect was the need to use two different *"on proximity event"* blocs. This is because with only the *"on event sensor detecting proximity"*, when the *horizontal proximity sensors* were triggered and the rotating direction was changed, the sensors were still triggered and so the FLAGS corresponding to the rotation direction would keep on switching so fast that the motors would not take the other values and the Thymio would keep turning in circle in one way.

However, implementing the *"on **no** proximity event"* bloc combined with the other one, this allowed the program to change the turning direction from clockwise to counterclockwise or vise-versa and hence not having the Thymio blocked in a loop of turning uniquely in one direction.



4 Educational Aspect

My goal for the educational aspect of this project is to establish three separate activities, one in each programming language. These three activities are designed to increase in difficulty and to teach/ challenge students with various, interesting approaches to programming.

VPL3 being the lowest programming level between the three, it will be the first activity proposed, then the Blockly one and finally the Python one.

VPL3

The VPL3 activity aims to teach the use of flags and introduce a challenging approach with logic operators. Students will receive a circuit featuring several black lines in various shapes and directions, along with an example of line-following code.

Their task is to use flags to enable the Thymio to follow these lines. The flags will signal to Thymio when it reaches the end of one line and needs to rotate to follow the next. Ultimately, the program should operate autonomously and continuously navigate through all the lines.

Blockly

An intriguing aspect that can be proposed for the Blockly activity is the introduction to sequential programming and understand its distinction from event-based programming. In this activity, the students will have to create a dance choreography for the Thymio while wearing the Doubled Costume.

The students will be provided a list of "dance moves" to choose from and use to build a complete choreography. To grasp the concept of sequential programming, their dance routine must run uninterrupted until it is finished. Although some movements can be based on sensor inputs, the sensor-dependent actions will only be checked and executed at the specific moments when the sensor values are requested.

Python

The Python activity aims to help students understand how the accelerometer sensors function in the Thymio. Students will use the *print()* function to identify which accelerometer sensor ([0], [1] or [2]) corresponds to specific inclination motion. For this activity, students will be given a Thymio equipped with the Costume 3.

This particular costume allows to visually display the motion depending on the degree of tilt of the Thymio. Students will be provided part of a Python code enabling them to read the accelerometer sensors values. Their task will be to complete the *accelerometer_ effect()* function to enable the costume move in specific ways according on the various inclinations.

5 Discussion

After implementing the same codes in the various programming languages, it is interesting to compare which are better suited for the different sensors and costume used.

I have reproduced the table of costumes and sensors pairing based on the estimated Table of subgroups of sensors & costumes previously mentioned. This table represents the final evaluation of the costume groups and sensors pairing. The last column indicates whether the specific programming language is compatible (or interesting enough) with the respective sensors.

COSTUMES GROUPS	A	B	C	Python	VPL3	Blockly
SENSORS						
prox.horizontal						
0 1 2 3 4	● ■	● ■	● ■	✓	✓	✓
5 6	● ■	● ■	● ■	✓	✓	✓
prox.ground						
0 1	● ■	● ■	● ■	✓	✓	✓
buttons	● ■	● ■	● ■	✓	✓	✓
microphone	● ■	● ■	● ■	✓	✓	✓
accelerometer						
0 1 2	● ■	● ■	● ■	✓	✗	✗

● Yes ● Improbable ● Can't be considered
● Probable ● No
○ Interactions with costumes □ Interactions with external observer

Figure 10: Final Table of subgroups of sensors & costumes

Buttons

The use of the Thymio's Buttons is mostly interesting while using the Python programming approach. This is because it allows to implement ***Threads*** and use the buttons to navigate through several custom programs that can be chosen by the programmer.

The navigation through the various programs is interesting for all the costume types. This is due to the fact that each program can be set to be used for a specific costume and so the user can switch programs when changing the Thymio's costume.

In the other programming languages, it could be interesting to use the buttons as well. For instance, in Blockly, the buttons could be used to integrate the concept of interruptions.

Horizontal Proximity Sensors

The *Horizontal Proximity Sensors* are the most versatile since their implementation in the three programming languages is quite similar and quite simple.

These sensors are particularly interesting to use with the Costume group A and with the Costume group C. In the case of the group A, the *Horizontal Proximity Sensors* can be used either for an automatic system but as well as for an interactive system with an external observer.

This allows for versatile options depending on the desired applications. In contrast, for the group C, the *Horizontal Proximity Sensors* can only be used for automatic systems because these costumes block the access of an external observer to the sensors and therefore they cannot trigger them without interfering with the costumes.

Ground Proximity Sensors

The *Ground Proximity Sensors* are comparable to the *Horizontal Proximity Sensors*. However, their placement on the Thymio was quite frustrating in the conceptualization of the backwards line following.

Aside from the Costume 3, these sensors can be used for any type of costumes. It can either be some black lines incorporated to the costume or it can also be interactive giving the external observer a list of black lines that they can chose and change at their leisure.

Accelerometer Sensors

The *Accelerometer Sensors* are mainly interesting to implement in the Python programming approach. This is due to the fact that in the VPL3 programming approach, the degree to which the robot is inclined is pretty vague and more complicated to put into place with the separation of the two different types of tilting angles.

On the other hand, in the Blockly programming approach it is not very interesting since the interval of the values that the sensors take need to be known. With Blockly, it is not possible to print the values in order to understand their effect depending on how one inclines the Thymio.

The Costume 3 is the most coherent costume to use with these sensors. It allows the external observer to understand how the various accelerometer sensors function and interact with each other to provide comprehensive motion understanding.

Microphone Sensors

The *Microphone Sensor* could be interesting to use regardless of the choice of the programming language. Nonetheless, in the context of an educational aspect, it can be difficult to put into place depending on the number of students participating in the activity and on the surrounding noise. The microphone's threshold can be adapted but if set too high, the sound required to trigger the sensors will need to be louder.

This sensor can be used for any costume types. All that's necessary is to combine the sound response with a motion that can be used correctly and that is tailored to the specific design of the chosen costume to prevent it from being damaged by inappropriate movements.

6 Conclusion

In conclusion, comparing these different programming languages is very interesting. Implementing the same programs across the various languages highlights the diverse approaches to programming topics at different levels. This demonstrates that these topics can be effectively introduced to students of various ages.

Additionally, pairing the costumes with different programs is very engaging. It enhances the illustrated concepts by accentuating Thymio's movements according to the chosen motions and can also be the centerpiece of the activity.

All documents related to my research, results, drawings, codes, etc., will be made available, especially to Ms. Léa Pereyre and Mr. Manuel Bernal Lecina, allowing everything that I have been able to develop during this project to be used. One of my personal goals was to create and to leave as much documents for the advancement of Léa Pereyre's project. I am already very delighted that this project has given the opportunity for the development of a new costume.

7 Acknowledgment

I really enjoyed this project and working with Ms. Léa Pereyre and Mr. Manuel Bernal Lecina was very pleasant. Being able to combine the knowledge I have acquired in my past semesters at EPFL with art and creativity was a big opportunity for me since I wish to develop a career combining robotics with artistic creativity. I wish to thank Professor Francesco Mondada for the chance to work as a part of the MOBOTS group.

8 Reference

[1] Thymio Cheat Sheet - given in the class *MICRO-452 : Basics of Mobile Robotics* tought by Professor F. Mondada in the fall semester

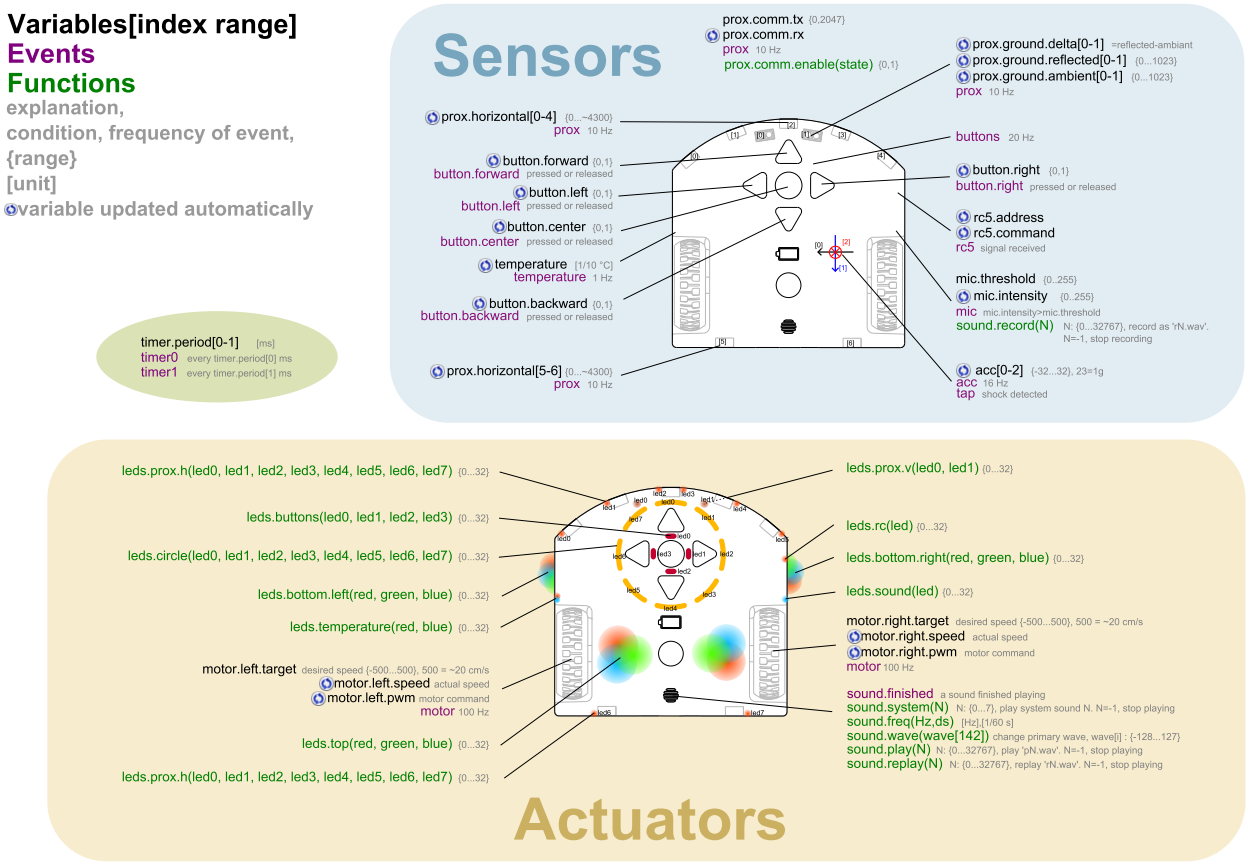


Figure 11: Schema of the Thymio Sensors

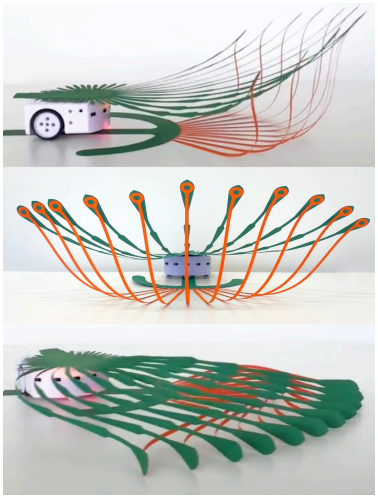
[2] <DÉ>CODAGE

Manuels pour l'enseignement de l'Éducation numérique (collection complète) published by "*la Direction générale de l'enseignement obligatoire et de la pédagogie spécialisée*" (DGEO) of the canton Vaud.

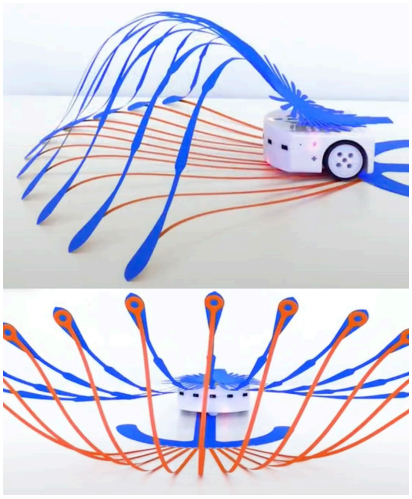
`https://decodage.edu-vd.ch`

9 Appendices

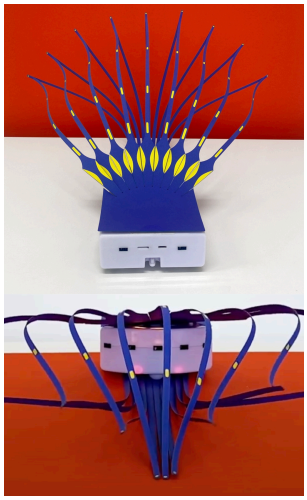
9.1 Costume images



(a) Costume 1



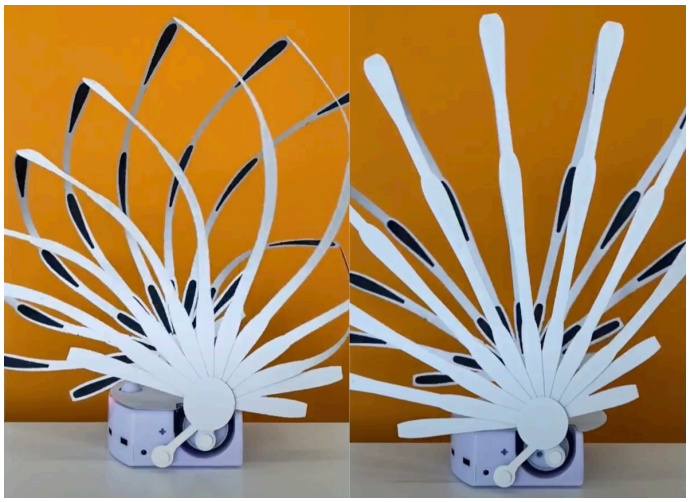
(b) Costume 4



(c) Costume 7

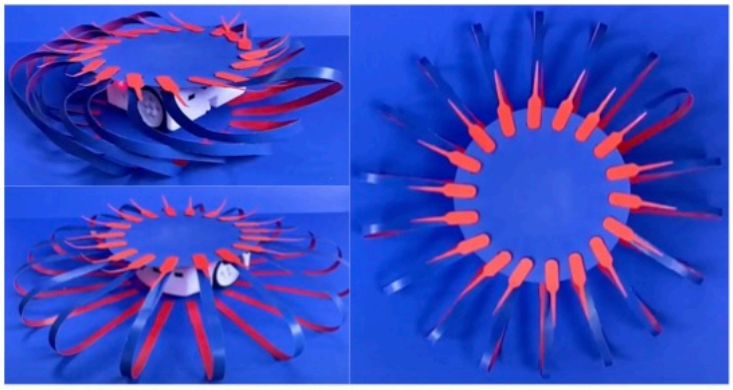
Appendix 1: Costume group A

(a) Costume 2

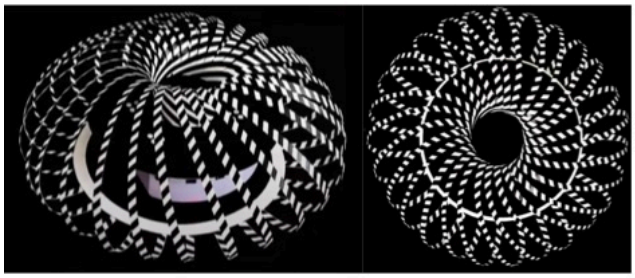


(b) Costume 3

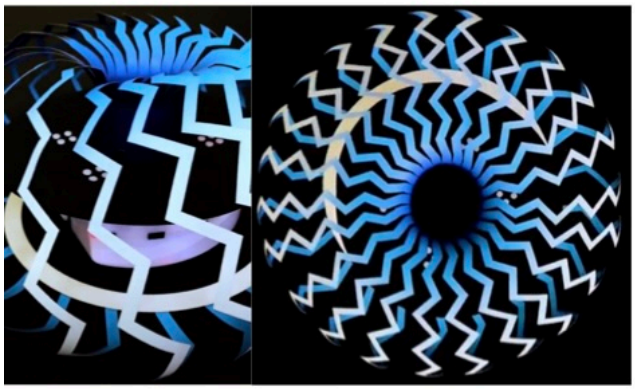
Appendix 2: Costume group B



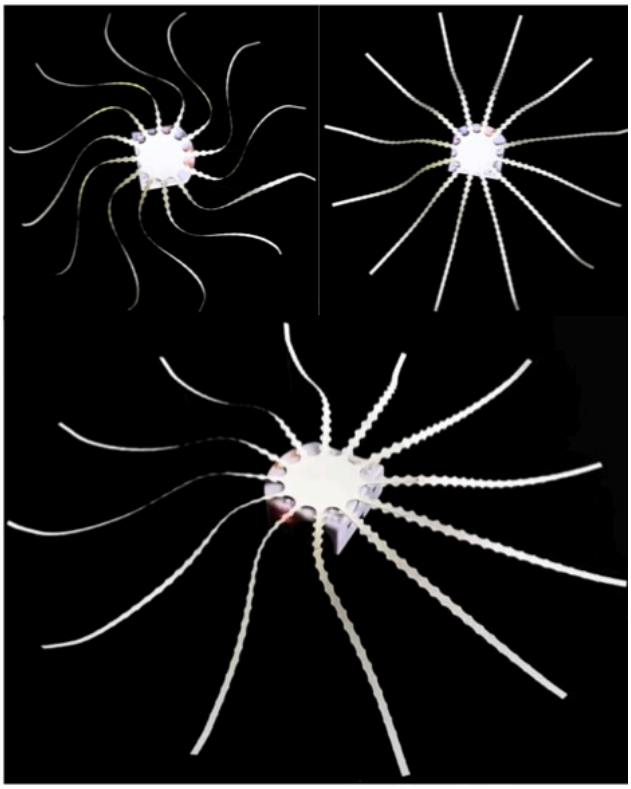
(a) Costume 5



(c) Costume 8



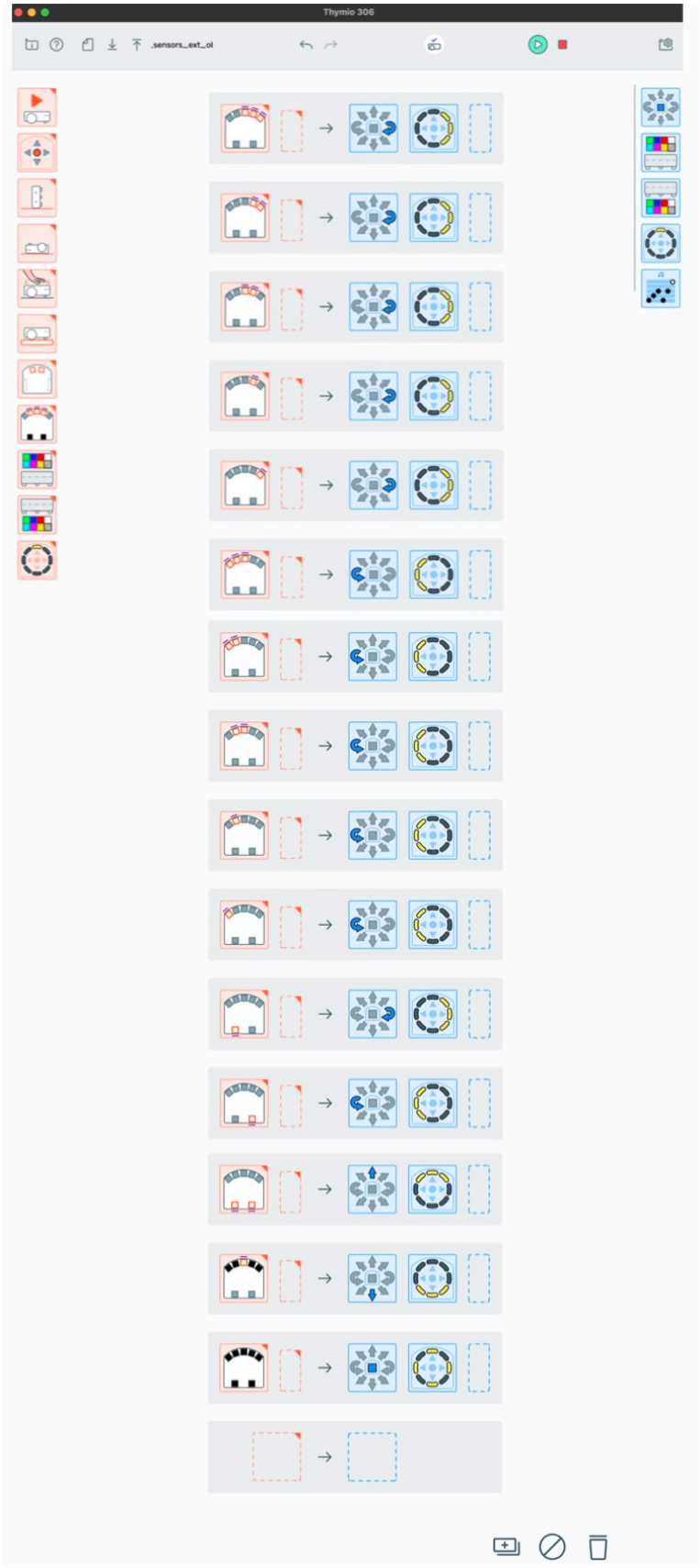
(d) Costume 9



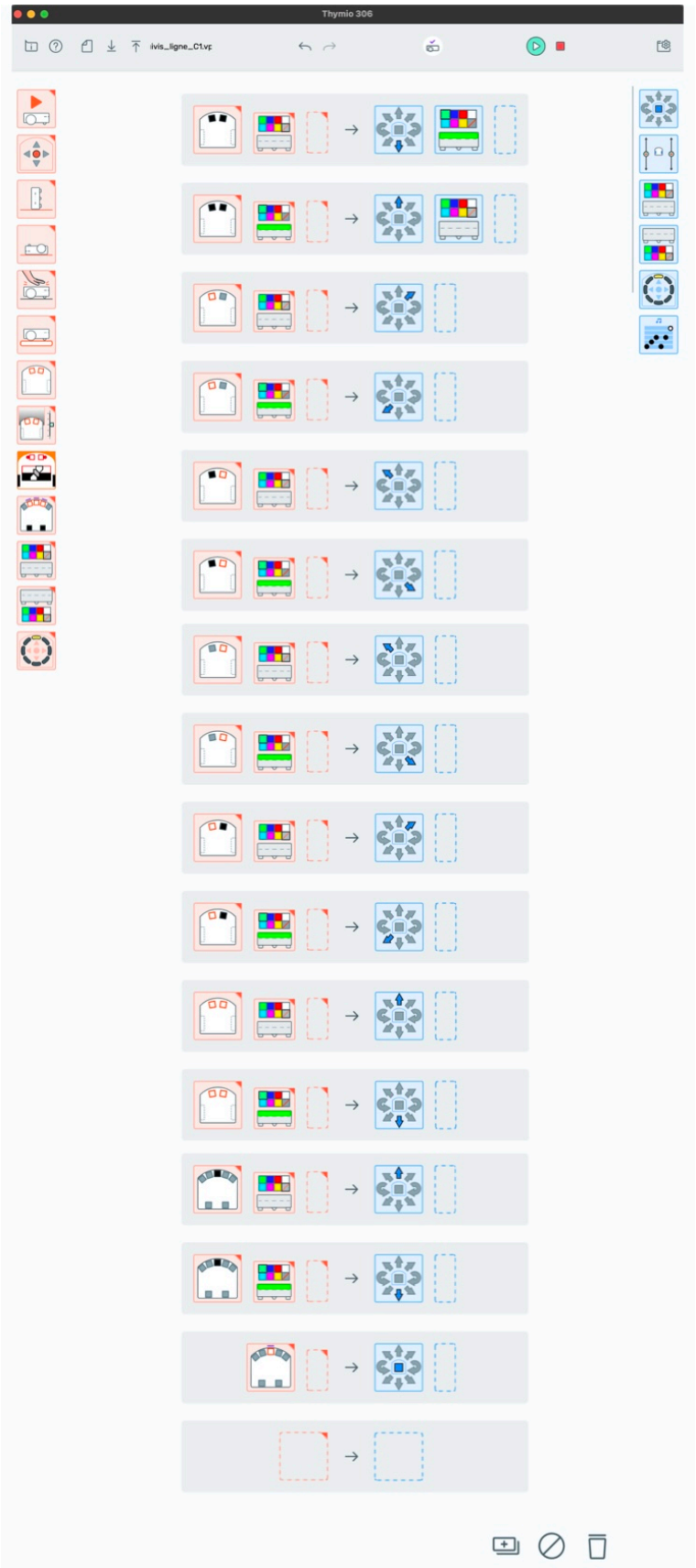
(b) Costume 6

Appendix 3: Costume group C

9.2 VPL3 code

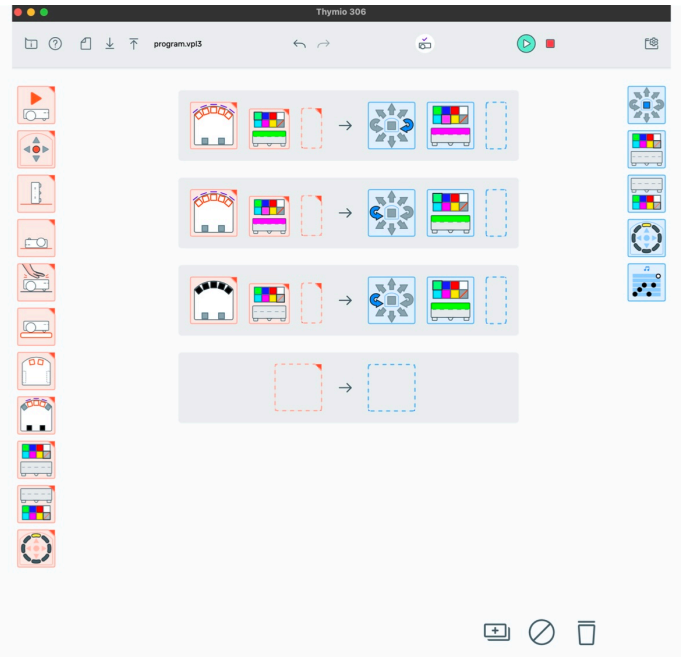


Appendix 4: VPL3 code for proximity sensors



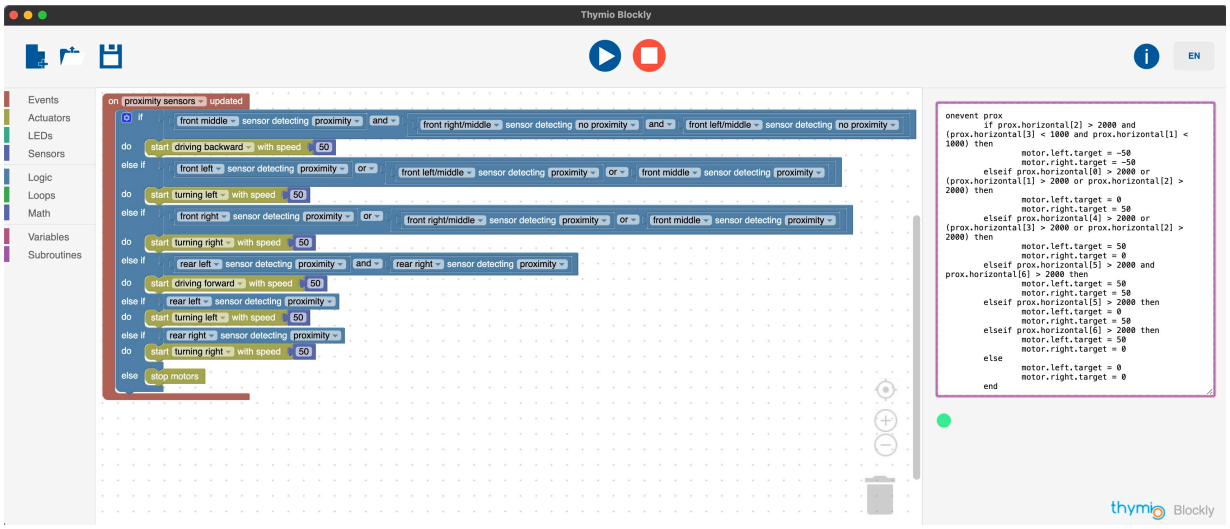
Appendix 5: VPL3 code for ground sensors

Appendix 6: VPL3 code for accelerometer sensors

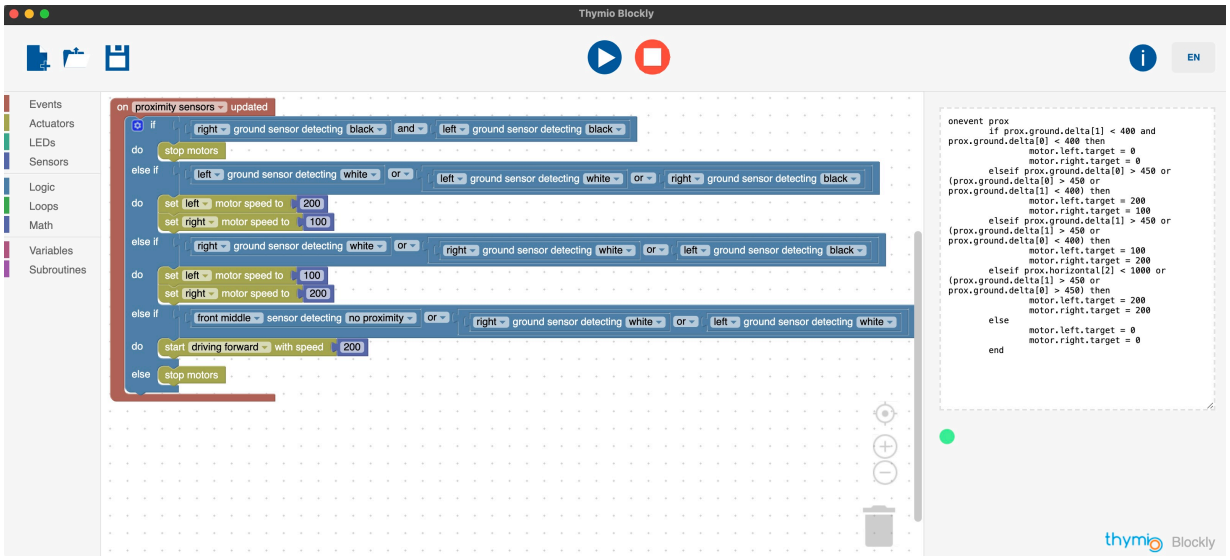


Appendix 7: VPL3 code for Costume 5

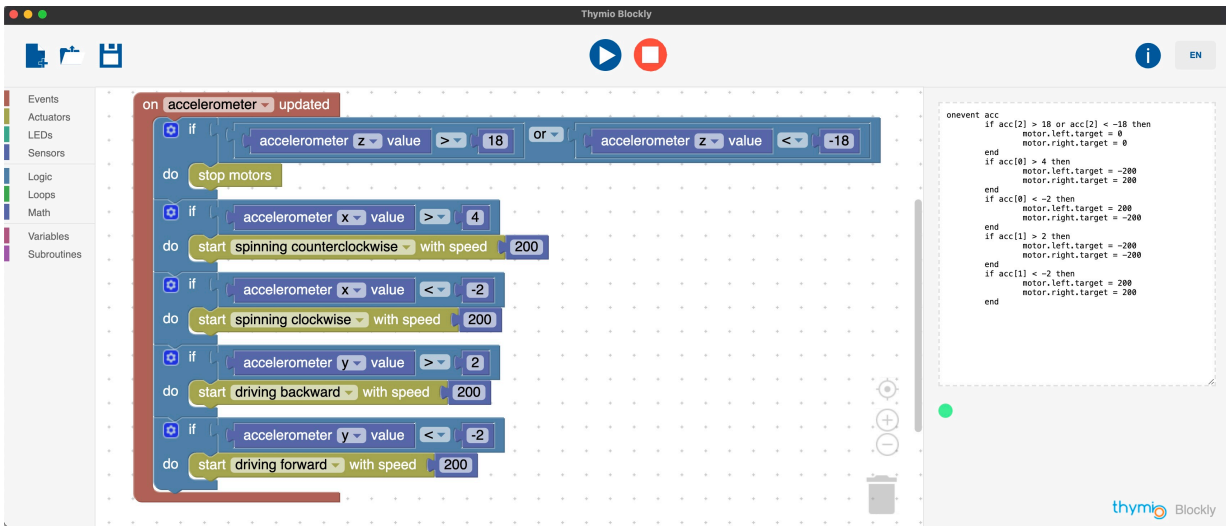
9.3 Blockly code



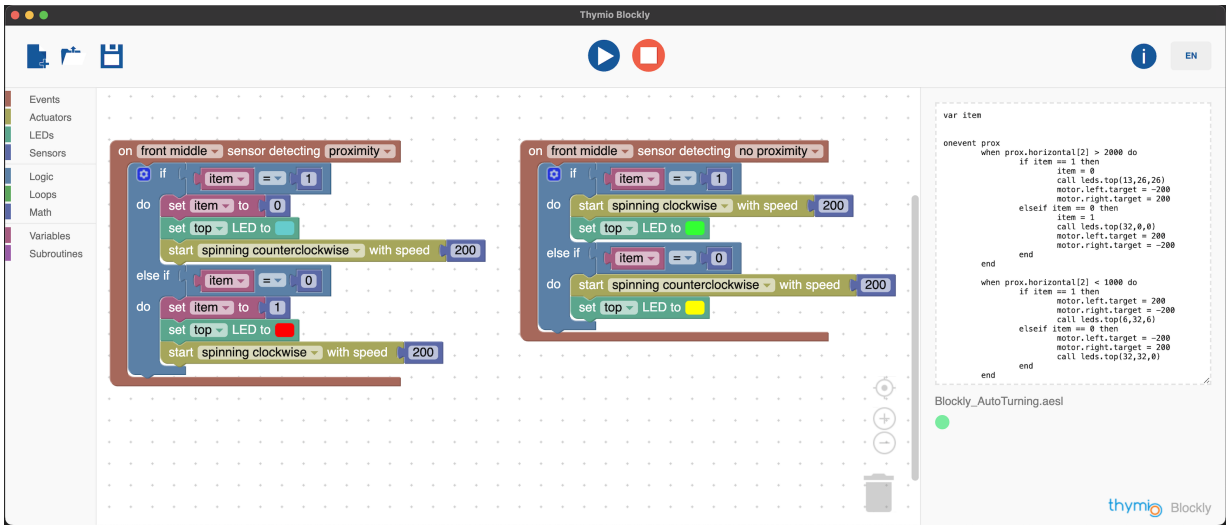
Appendix 8: Blockly code for proximity sensors



Appendix 9: Blockly code for forward line following



Appendix 10: Blockly code for accelerometer



Appendix 11: Blockly code for the automatic rotation program

VPL3 Session :

Addressing the FLAGS approach

	SUMMARY	The students must program the Thymio an autonomous line following code so that it will go round and round the given line pattern.
	MATERIAL	• Line following code • Line to follow circuit • Thymio robot • Computer with «Thymio Suite»

For this session, the constraints given to the students are such that the Thymio must :

- move forward following a black line until it reaches the perpendicular black line
- as soon as it reached the perpendicular black line, move backward following the same black line.

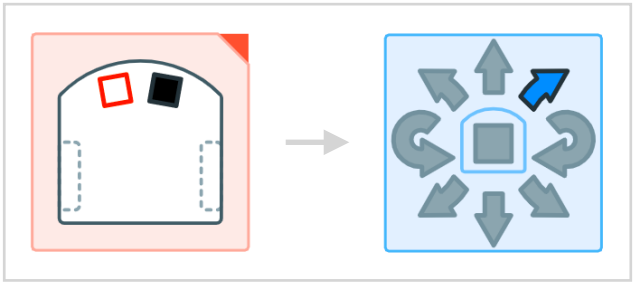
To reach this goal, the students are given part of the code that they will need to complete. They are challenged to think about how to approach the problem while using flags so the robot knows if it's in the mode moving forward or the mode moving backward.

The students will use «Thymio Suite» and code with «VPL3» programming language.

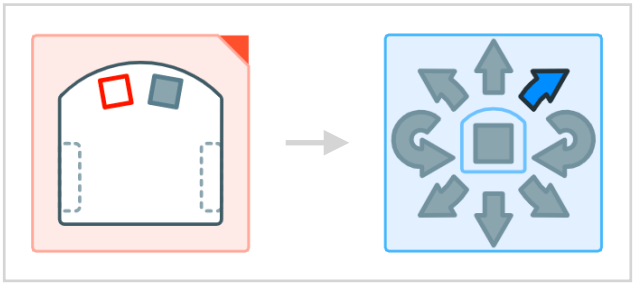
There are three cases for the **ground sensors** :

SENSOR'S COLOR	gray	white & red	black
THYMIO'S MOVEMENT	The Thymio doesn't take into account any information perceived by the sensors.	The Thymio reacts as soon as the sensors perceive some white or light underneath it.	The Thymio reacts as soon as the sensors perceive some black or dark underneath it.

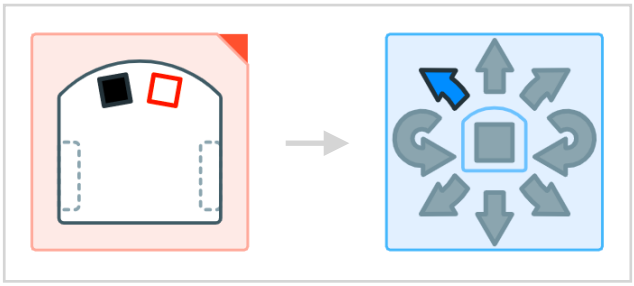
VPL3 code for following a line :



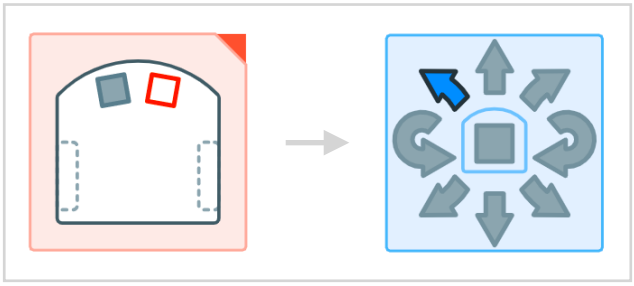
Thymio detects white on the left ground sensor and black on the right ground sensor : turn lightly to the right



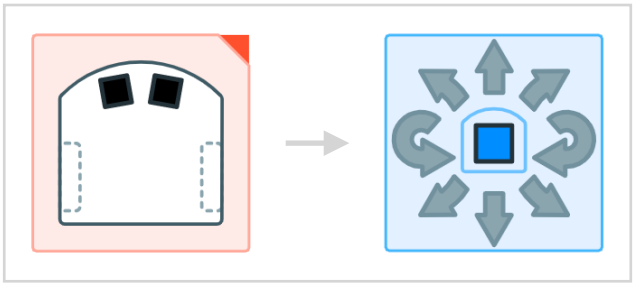
Thymio detects white on the left ground sensor and no matter what the right ground sensor senses : turn lightly to the right



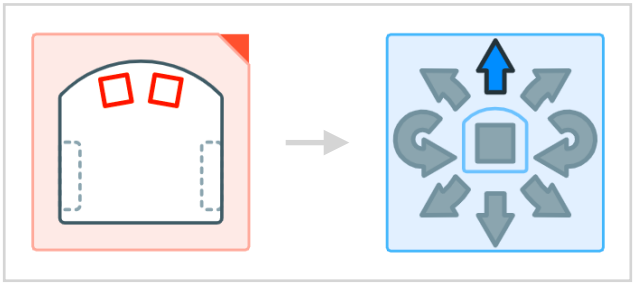
Thymio detects white on the right ground sensor and black on the left ground sensor : turn lightly to the left



Thymio detects white on the right ground sensor and no matter what the left ground sensor senses : turn lightly to the left



Thymio detects black on the left and on the right ground sensors : stop



Thymio detects white on the left and on the right ground sensors : go forward

Challenge I

Starting position

The Thymio is placed on one line with the two ground sensors on either side of the black line after the starting black rectangle of the chosen line.

Print the last page in format A3

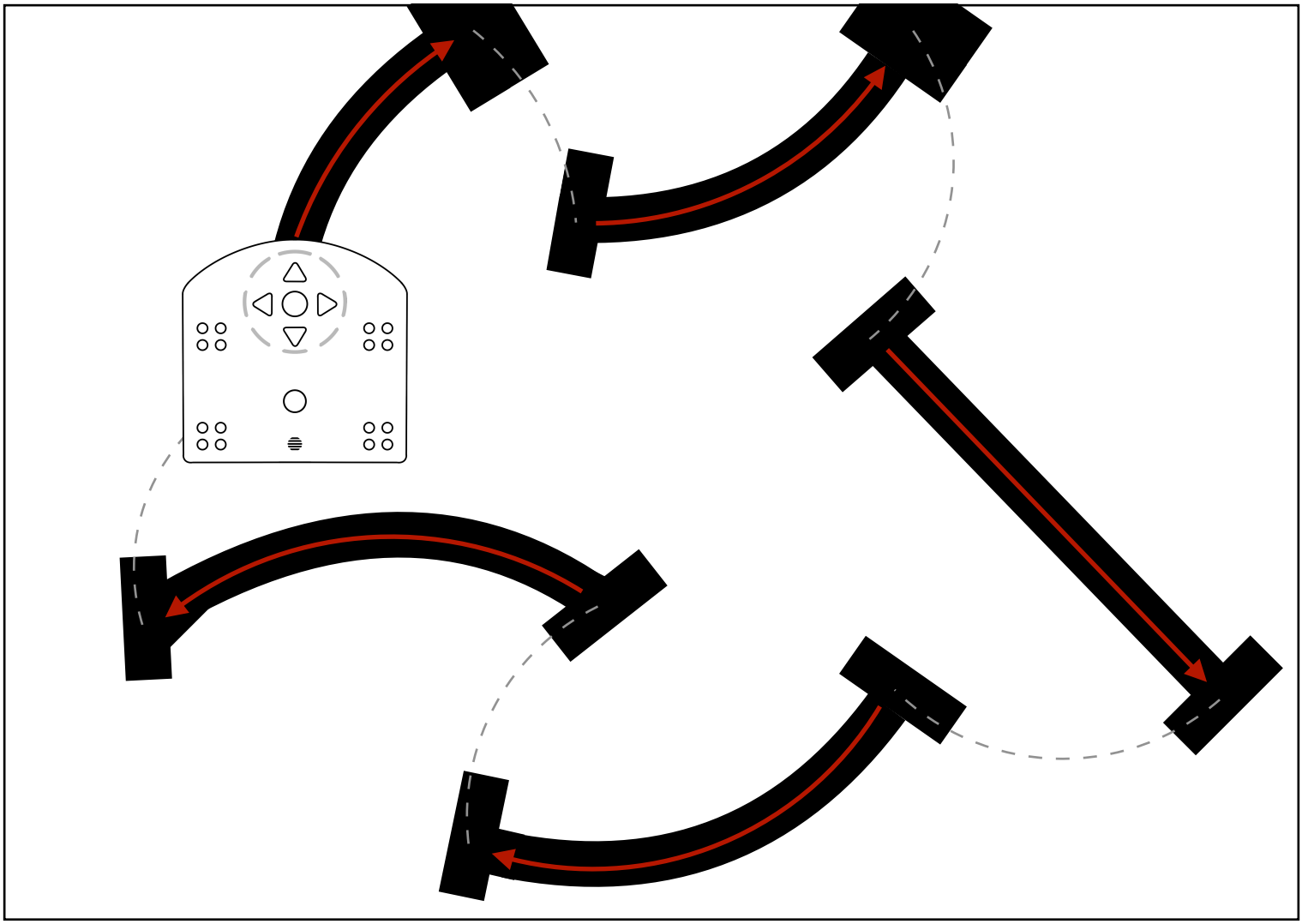
Goal

The Thymio should follow the lines until the black rectangle and then rotate on itself until it reaches the starting black rectangle of the next line. It should be able to follow every line one after the other several times in a row.

How

Use the ground sensors and the two wheels.

Need a hint ? Have a look at the next page !



Hint :

Top color state

Set top color

Use the **Top color state** as a condition in the event blocks and use the **Set top color** in the action blocks as FLAGS.

Having the Thymio set in a certain color as a condition can allow adding some actions for similar events.

For instance, if the ground sensors are detecting black and the Thymio is green, the action can be stopping the wheels but if the Thymio is pink for the same sensor detection, another action can be executed.

Challenge II

If the first challenge was to easy, try coming up with your own pattern of lines to follow.

You can use some of the lines given on the next page.:

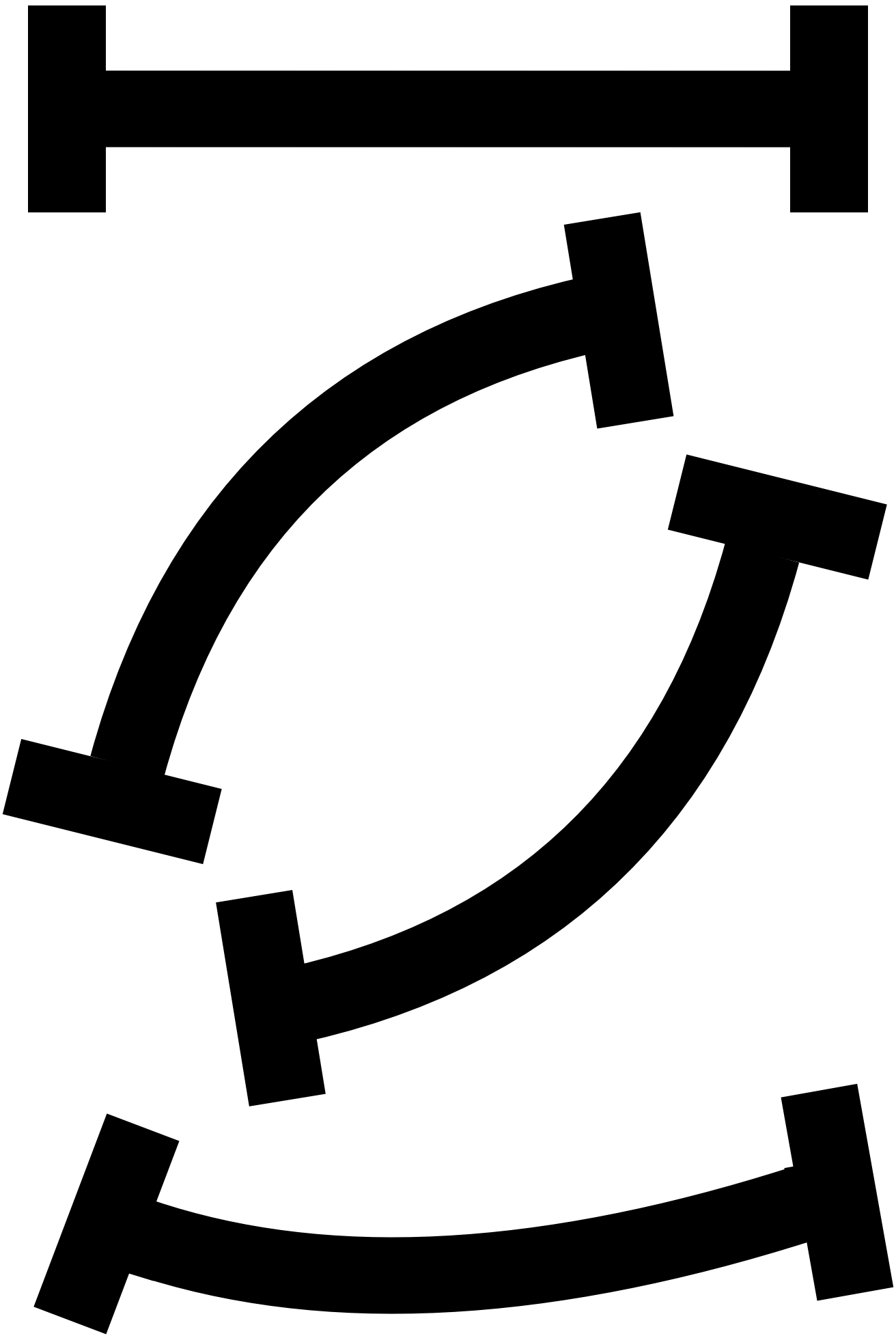
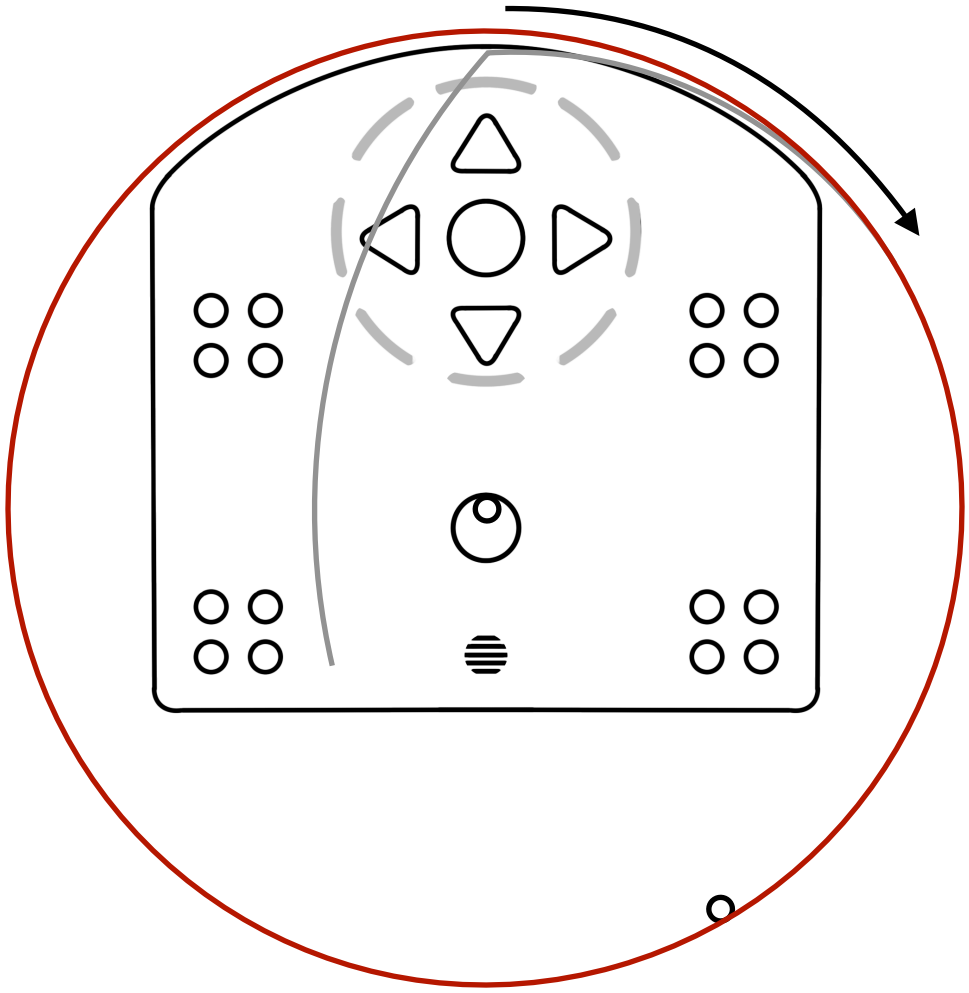
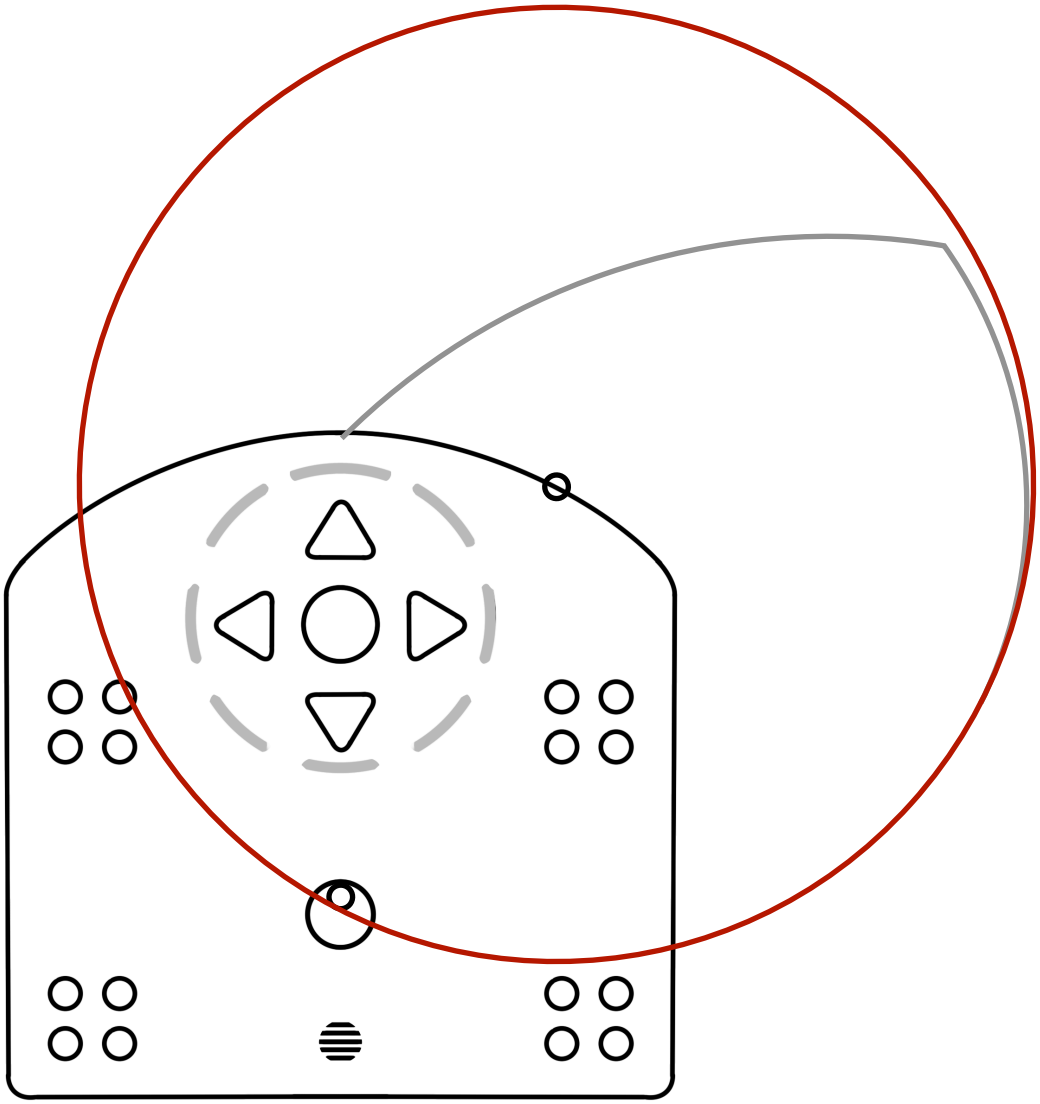
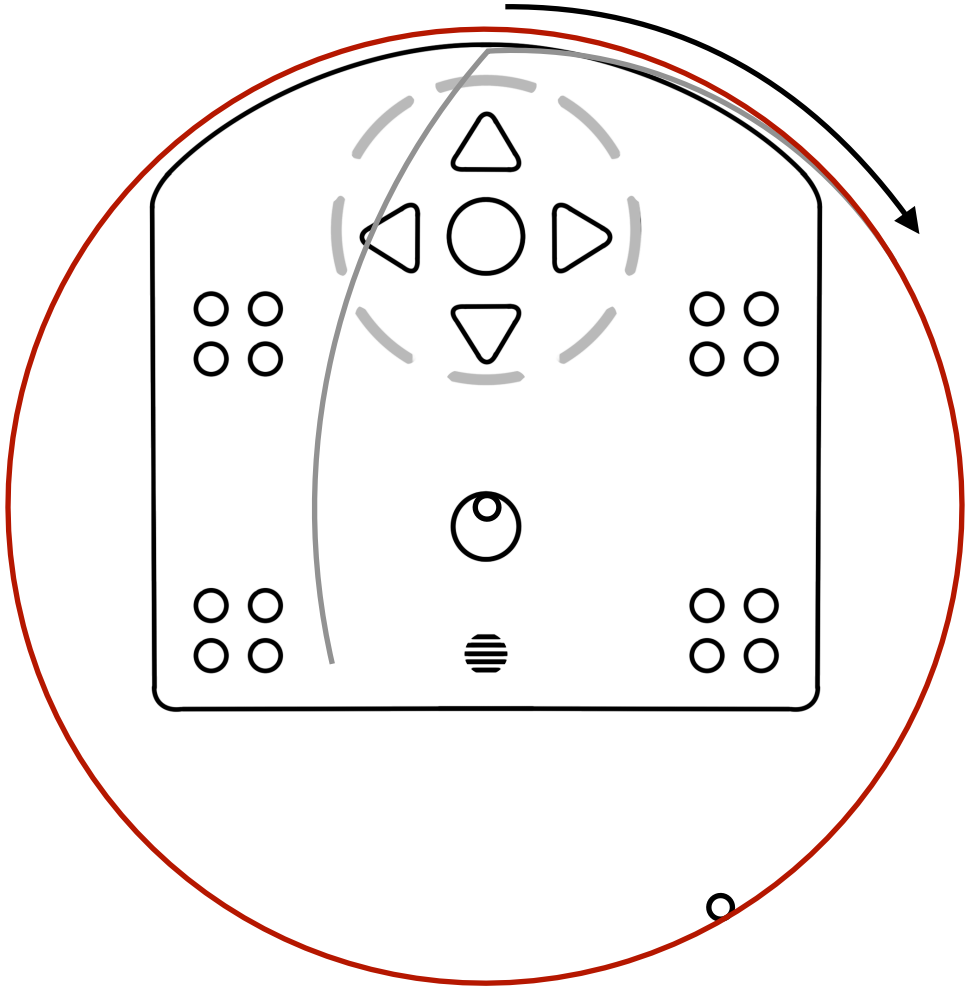
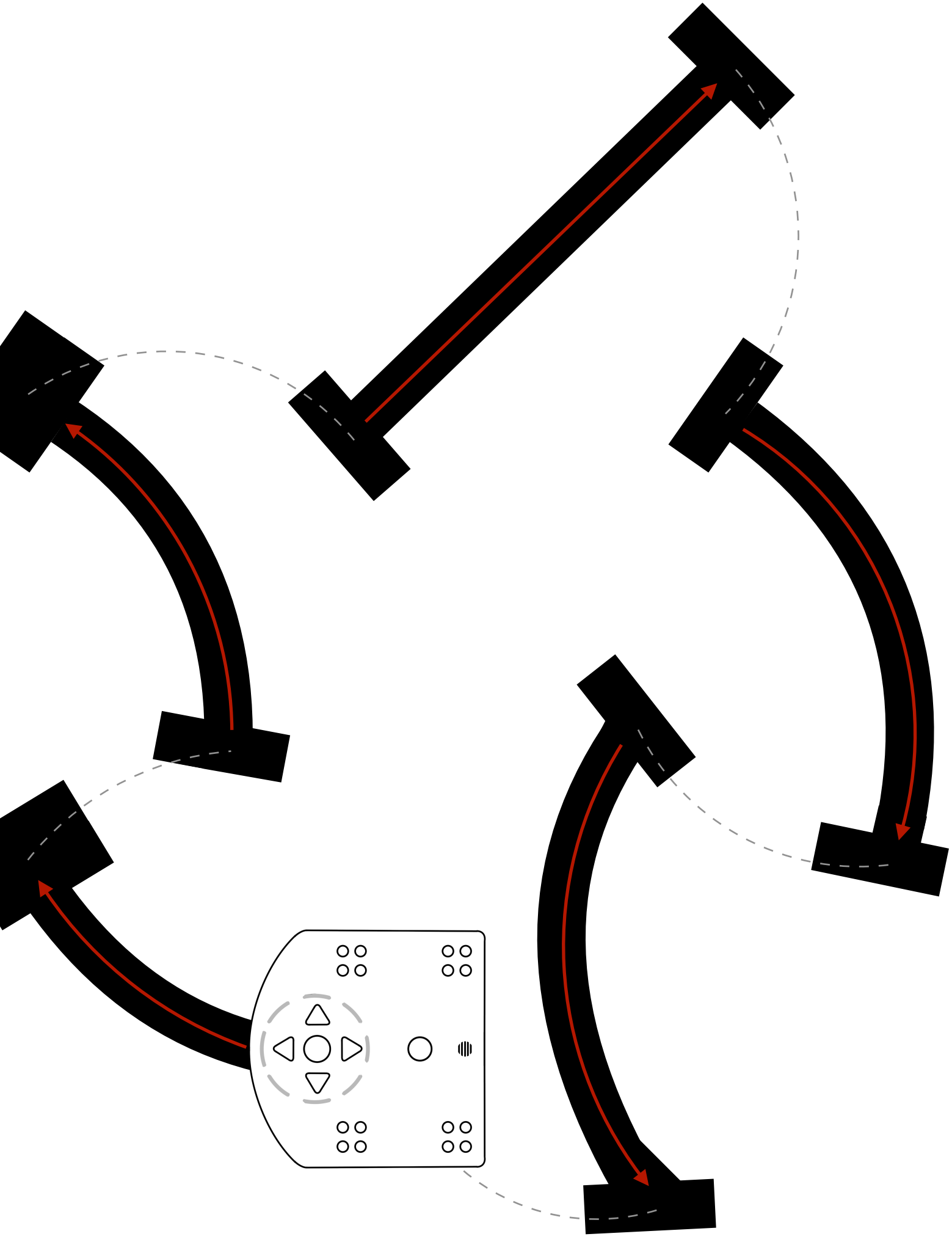


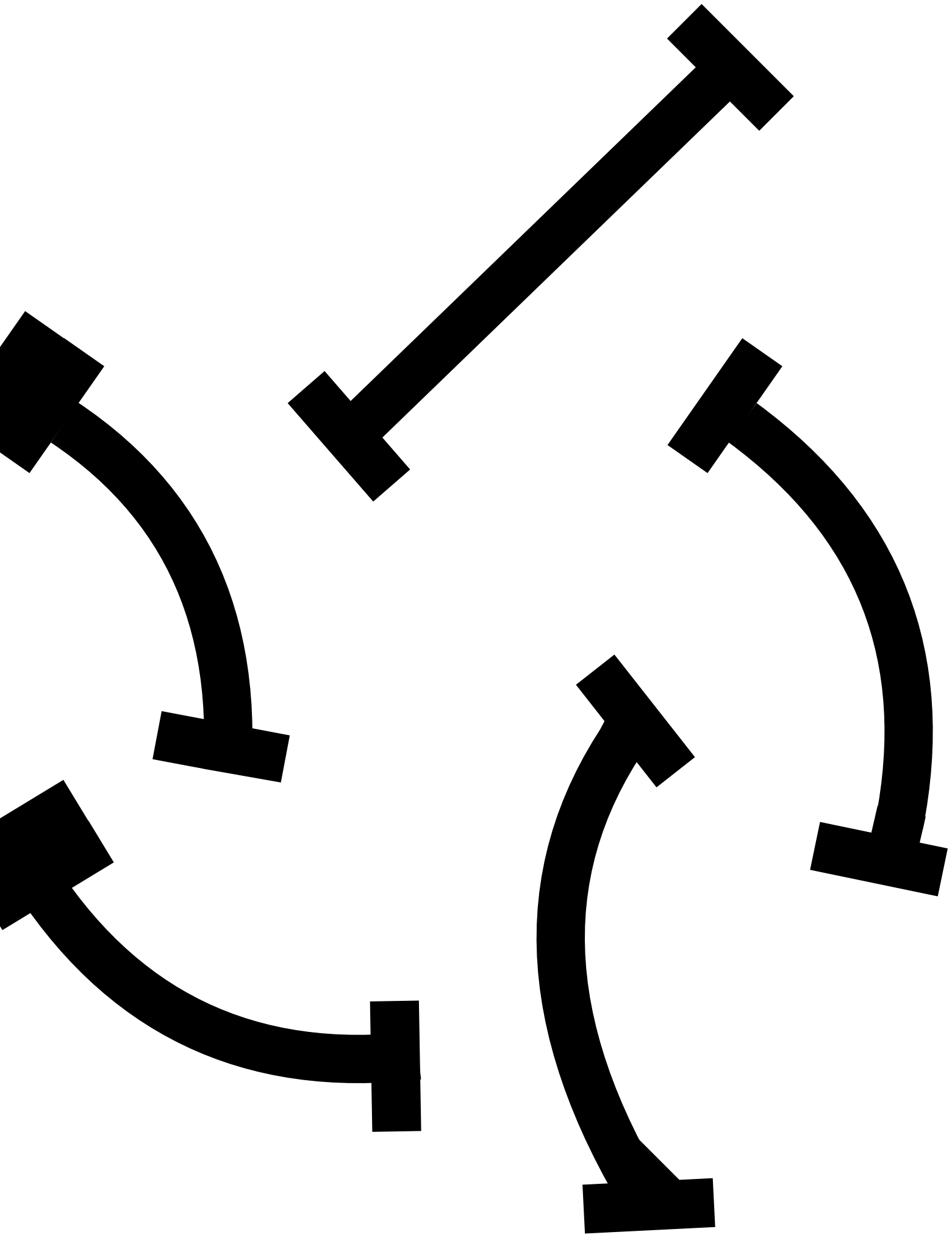
Illustration of the circle the Thymio draws when turning on itself :



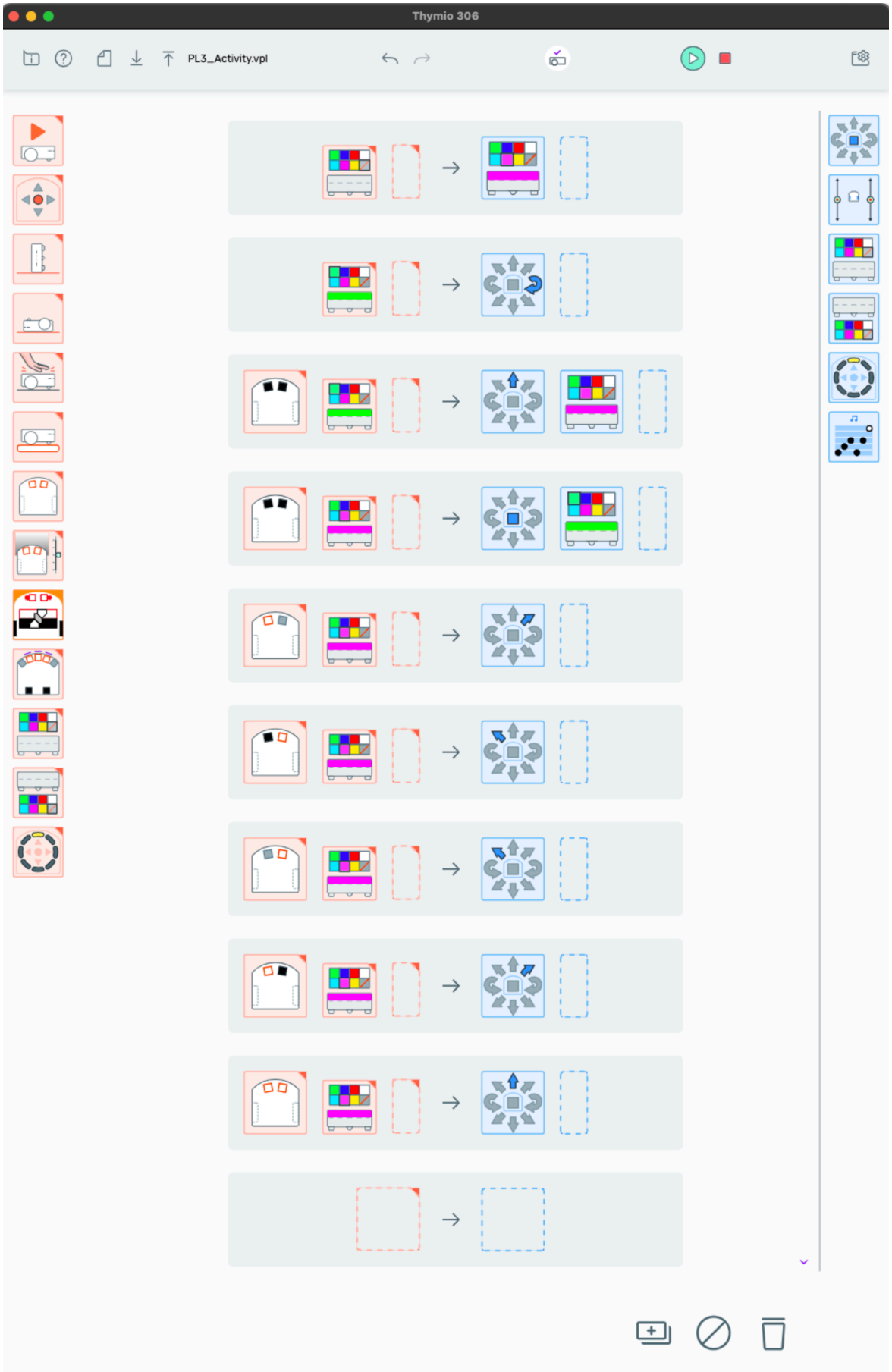








Solution :



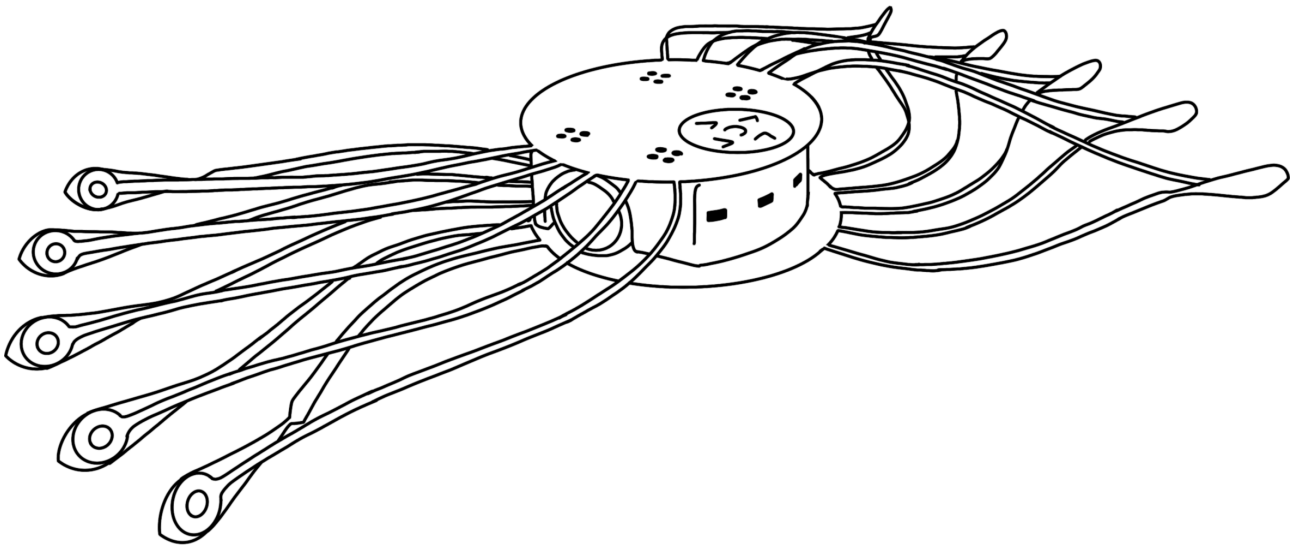
BLOCKLY Session :

Addressing sequential programming

	SUMMARY	The students must program a dance choreography for the Thymio based on a list of possible “dance moves”.
	MATERIAL	• List of dance moves • Thymio robot • Computer with «Thymio Suite»

An interesting aspect that can be brought up in this exercise is an introduction to sequential programming and understand the difference with event based programming.

The activity is to propose to the students to create a dance choreography for the Thymio while wearing the following costume :



The students are given a list of “dance moves” from which they can chose and create a full choreography. To understand the concept of sequential programming, their dance routine cannot be interrupted until it’s finished.

If the exercise seems to simple for some students, they can chose movements that depend on some sensors. However, the motion dependent on a sensor will only be checked and executed at the specific moment the request to check the sensor value is asked, adding an extra level of difficulty.

Challenge I

Starting position

The Thymio is dressed with the given costume.

Goal

The Thymio should follow a dance choreography created by the student.

How

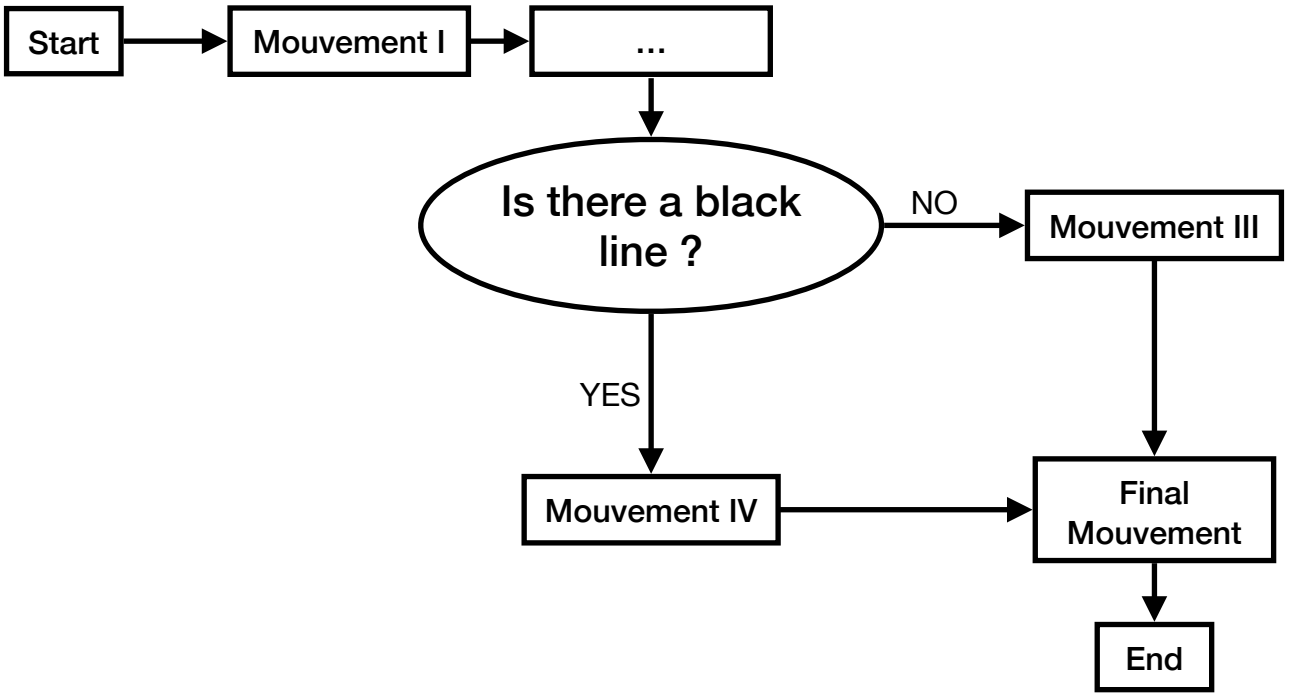
Use the “onvent timer” events with variables for the conditions.

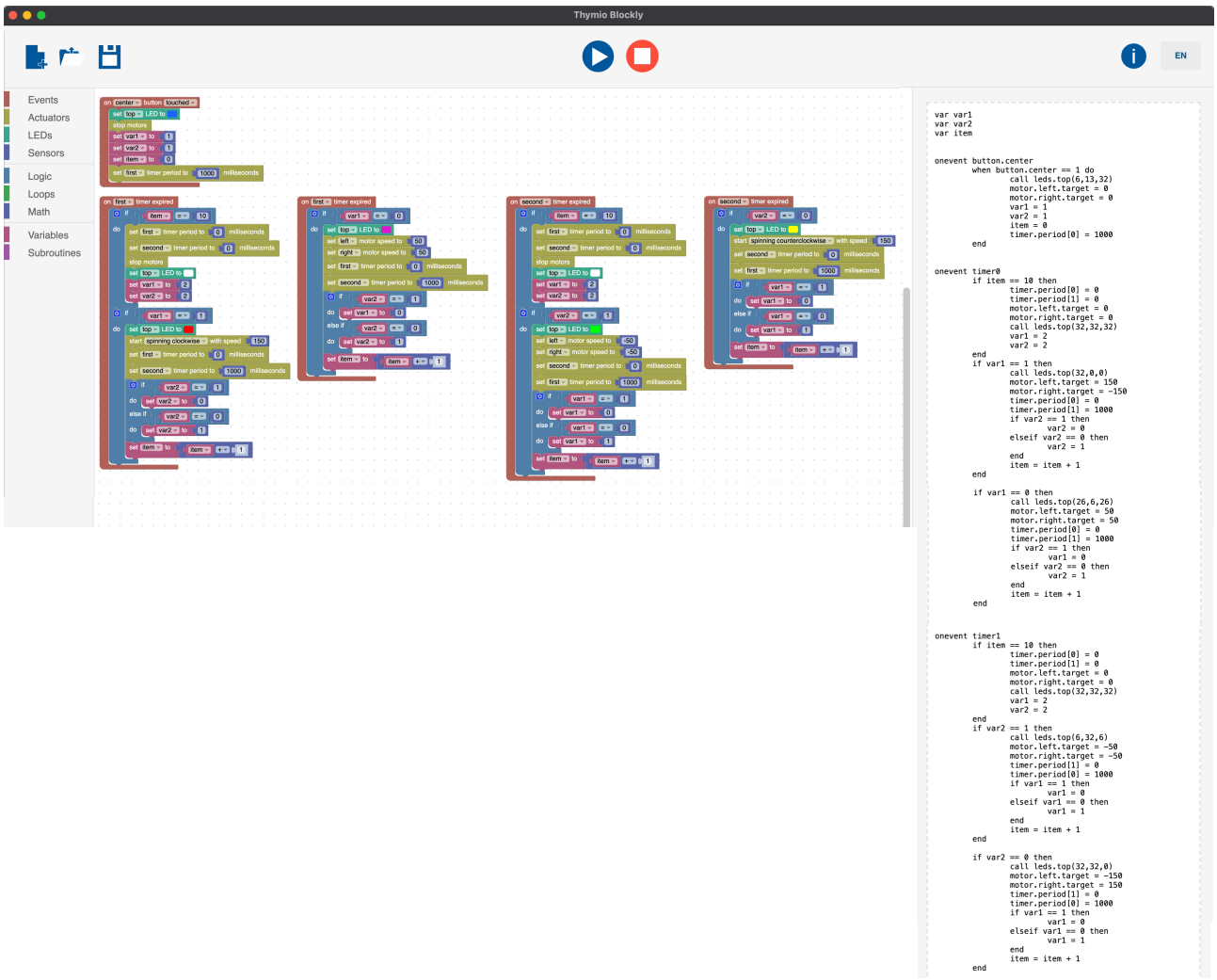
Need a hint ? Have a look at the example given on the next page !

Challenge II

If the first challenge was too easy, try creating a dance choreography that has movements dependent on some sensors (the ground proximity sensors for instance).

You have to keep in mind that the motion sensor dependent can only be executed at the specific moment the request to check the sensor value is asked !






```
var var1
var var2
var item


onevent button.center
	when button.center == 1 do
		call leds.top(6,13,32)
		motor.left.target = 0
		motor.right.target = 0
		var1 = 1
		var2 = 1
		item = 0
		timer.period[0] = 1000
	end


onevent timer0
	if item == 10 then
		timer.period[0] = 0
		timer.period[1] = 0
		motor.left.target = 0
		motor.right.target = 0
		call leds.top(32,32,32)
		var1 = 2
		var2 = 2
	end
	if var1 == 1 then
		call leds.top(32,0,0)
		motor.left.target = 150
		motor.right.target = -150
		timer.period[0] = 0
		timer.period[1] = 1000
		if var2 == 1 then
			var2 = 0
		elseif var2 == 0 then
			var2 = 1
		end
		item = item + 1
	end

	if var1 == 0 then
		call leds.top(26,6,26)
		motor.left.target = 50
		motor.right.target = 50
		timer.period[0] = 0
		timer.period[1] = 1000
		if var2 == 1 then
			var1 = 0
		elseif var2 == 0 then
			var2 = 1
		end
		item = item + 1
	end


onevent timer1
	if item == 10 then
		timer.period[0] = 0
		timer.period[1] = 0
		motor.left.target = 0
		motor.right.target = 0
		call leds.top(32,32,32)
		var1 = 2
		var2 = 2
	end
	if var2 == 1 then
		call leds.top(6,32,6)
		motor.left.target = -50
		motor.right.target = -50
		timer.period[1] = 0
		timer.period[0] = 1000
		if var1 == 1 then
			var1 = 0
		elseif var1 == 0 then
			var1 = 1
		end
		item = item + 1
	end

	if var2 == 0 then
		call leds.top(32,32,0)
		motor.left.target = -150
		motor.right.target = 150
		timer.period[1] = 0
		timer.period[0] = 1000
		if var1 == 1 then
			var1 = 0
		elseif var1 == 0 then
			var1 = 1
		end
		item = item + 1
	end
```

PYTHON Session :

Addressing the accelerometer sensors

	SUMMARY	The students must program the Thymio in order to make the costume move back and forth depending on the induced inclination.
	MATERIAL	• Part of a Python code that needs to be completed • Thymio robot • Computer with «Thymio Suite»

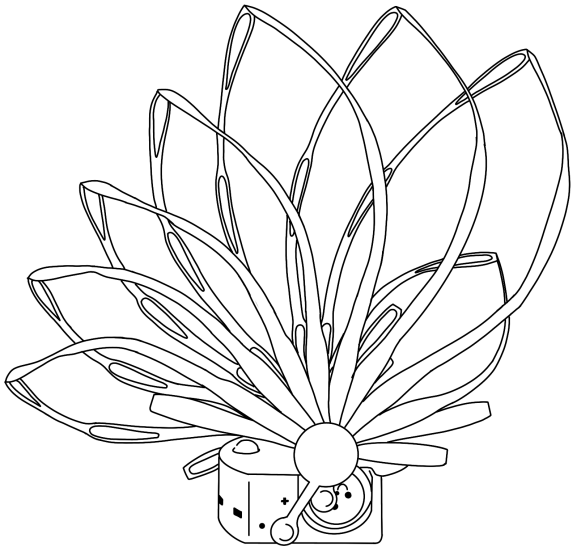
An interesting aspect that can be brought up in this exercise is pushing the students to understand how do the accelerometer sensors function in the Thymio.

For the proposed exercise of the Python session, the students must :

- use the ***print()*** function in order to observe the which accelerometer sensor ([0], [1] or [2]) correspond to which inclination motion.
- complete the ***accelerometer_effect()*** function with the sensor values they have observed beforehand .

To proceed in this activity, the students will be given a Thymio with the following costume :

This costume allows the students to observe visually the provided motion depending on the amount the Thymio is tilted.

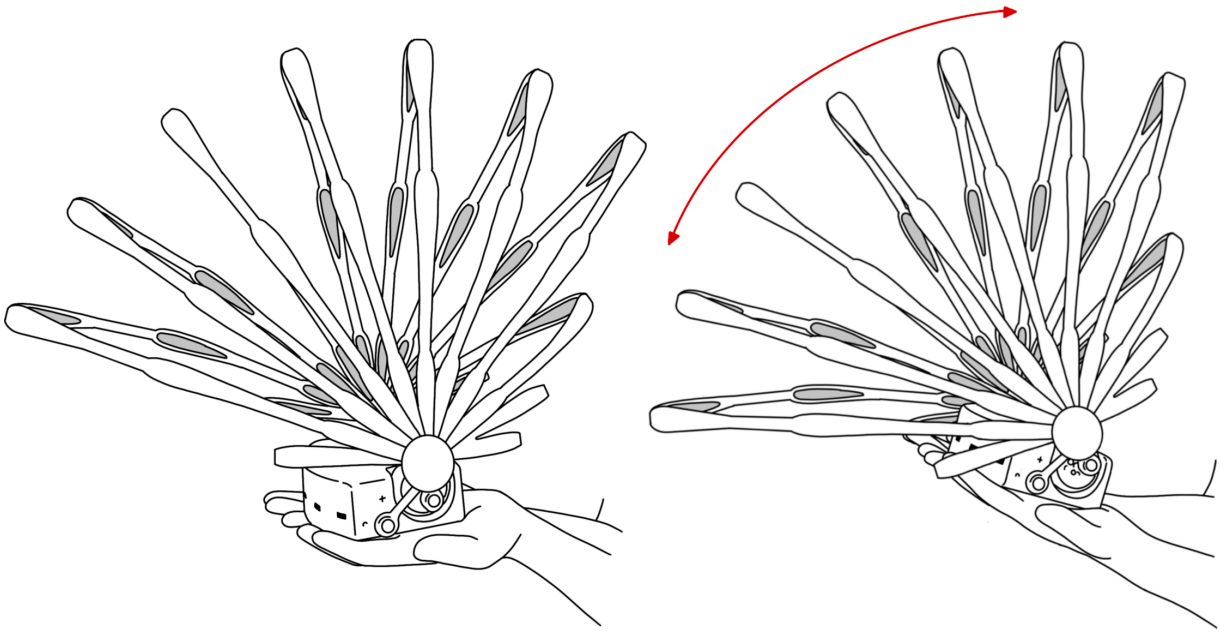


They are provided with a part of a python code enabling them to read the accelerometer sensors values and the ***accelerometer_effect()*** function they have to complete in order to provide specific costume movements depending on the different inclinations.

Challenge I

Goal

The Thymio should move the costume back and forth when the student holding the robot inclines it a certain amount either towards the front or the back, as can be seen below.



How

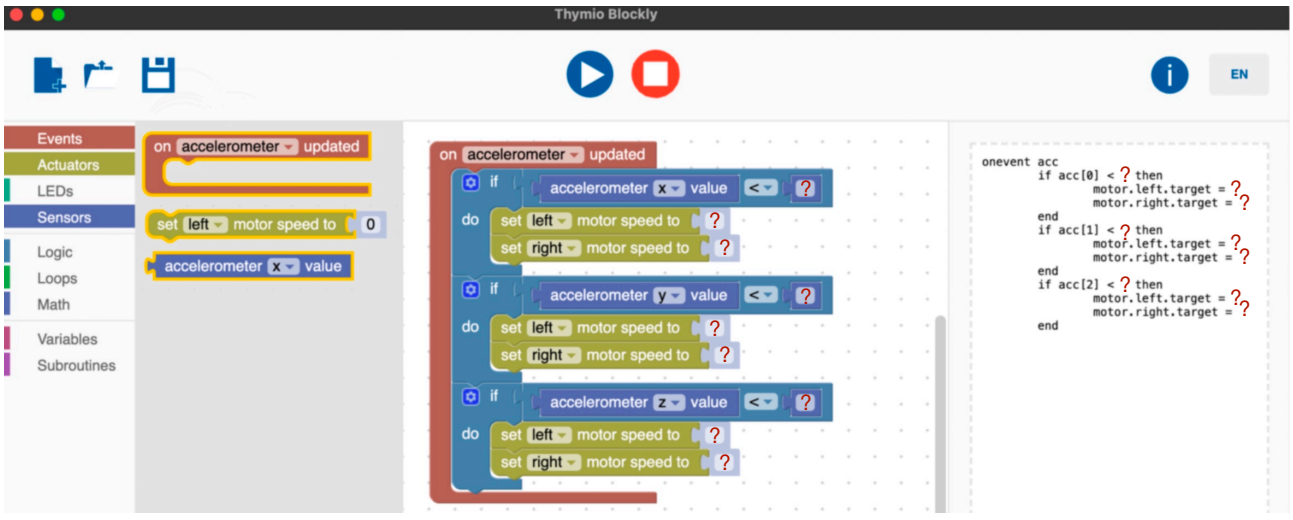
Use the accelerometer sensors and the two wheels.

To complete the ***accelerometer_effect()*** function, try writing the conditions is this form :

```
if accel[0] > ?? :

    # on the side
    robot.setSpeedLeft(0, node)
    robot.setSpeedRight(motor_speed, node)
```

Need a hint ? Try using blockly to understand how to code the function conditions



The given Python code to fill :

activityPythonToComplete.py > ...

```python
from tdmclient import ClientAsync, aw

import classes
from classes import Thymio

import numpy as np


client = ClientAsync()
node = aw(client.wait_for_node())

aw(node.lock())
aw(node.wait_for_variables())

robot = Thymio()

def update_sensors_data(robot, node):

    # get button values
    robot.getCenterButton(node)

def stop_program(robot, node, motor_speed=0) :

    robot.setLEDTop(node, [0,0,0])

    robot.setSpeedLeft(motor_speed, node)
    robot.setSpeedRight(motor_speed, node)

def accelerometer_effect(robot, node, motor_speed=100) :
    accel = list(node["acc"]) + [0]

    print(accel[0])

    print(accel[1])

    print(accel[2])

while(1) :

    update_sensors_data(robot, node)

    accelerometer_effect(robot, node, motor_speed=100)

    if (robot.button_center) :

        print(robot.button_center)
        stop_program(robot, node, motor_speed=0)
        aw(node.unlock())
        break
```

activityPythonToComplete.py > ...

```python
from tdmclient import ClientAsync, aw

import classes
from classes import Thymio

import numpy as np


client = ClientAsync()
node = aw(client.wait_for_node())

aw(node.lock())
aw(node.wait_for_variables())

robot = Thymio()

def update_sensors_data(robot, node):

    # get button values
    robot.getCenterButton(node)

def stop_program(robot, node, motor_speed=0) :

    robot.setLEDTop(node, [0,0,0])

    robot.setSpeedLeft(motor_speed, node)
    robot.setSpeedRight(motor_speed, node)

def accelerometer_effect(robot, node, motor_speed=100) :
    accel = list(node["acc"]) + [0]

    print(accel[0])

    print(accel[1])

    print(accel[2])


    # GAUCHE
    if accel[0] > ??: #Thymio is blue when placed on one of its sides
        robot.setLEDTop(node, [0,0,32])
        # on the side
        robot.setSpeedLeft(0, node)
        robot.setSpeedRight(motor_speed, node)

    # DROITE
    if accel[0] < ??: #Thymio is blue when placed on one of its sides
        robot.setLEDTop(node, [0,0,32])
        # on the side
        robot.setSpeedLeft(motor_speed, node)
        robot.setSpeedRight(0, node)

    # DERRIERE
    if accel[1] > ??: #Thymio is red when placed on its front or backside
        robot.setLEDTop(node, [32,0,0])
        # on back
        robot.setSpeedLeft(-motor_speed, node)
        robot.setSpeedRight(-motor_speed, node)

    # DEVANT
    if accel[1] < ??: #Thymio is red when placed on its front or backside
        robot.setLEDTop(node, [32,0,0])
        # on front
        robot.setSpeedLeft(motor_speed, node)
        robot.setSpeedRight(motor_speed, node)

    if accel[2] > ?? or accel[2] < ??: #Thymio is green when placed on its wheels or upside-down
        robot.setLEDTop(node, [0,32,0])
        # horizontal
        robot.setSpeedLeft(0, node)
        robot.setSpeedRight(0, node)


while(1) :

    update_sensors_data(robot, node)

    accelerometer_effect(robot, node, motor_speed=100)

    if (robot.button_center) :

        print(robot.button_center)
        stop_program(robot, node, motor_speed=0)
        aw(node.unlock())
        break
```