ROBOESL PROJECT

ROBOTICS-BASED LEARNING INTERVENTIONS FOR PREVENTING SCHOOL FAILURE AND EARLY SCHOOL LEAVING

Erasmus+ 2015-1-IT02-KA201-015141

Output 1: Curricula for 10 exemplary interdisciplinary robotics projects

Curriculum 3: The desert scout

Lead Partner: UNIPD (IT)

Authors: Michele Moro, Francesca Agatolio, Emanuele Menegatti (UNIPD)

Contributions

Dimitris Alimisis (EDUMOTIVA[1]) (Chapters 2 and 3)

Linda Daniela (University of Latvia) (Chapter 5)

Circulation: Public

Version: Final

Stage:

Date: March 31, 2017

[1] EDUMOTIVA stands for 'European Lab for Educational Technology

Declaration

This report has been prepared in the context of the ROBOESL project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Abstract

This document contains the description of the Curriculum n. 3, entitled 'The desert scout' which is part of the Intellectual Output 1 (Curricula for 10 exemplary interdisciplinary robotics projects) developed and tested in teacher training courses within the context of the ERASMUS+ ROBOESL project. In this Curriculum the robot is requested to pivot on each vertex of a regular polygon in order to follow its contour. A color sensor is used to detect color codes on each vertex which simulates a collection of corings. Together with straight line commands introduced in the previous curricula, here we have the possibility to reproduce the basic four commands of a Logo floor turtle.

Chapter 1: **Short description and scenario (O1.1)**

(source: nasa.gov)

Due to prohibitive conditions for humans, an autonomous robot is requested to visit the vertices of a regular n-sided polygon where to make a coring followed by a measure of quality of the extracted sample, emulated by a reading with the light sensor, in order to discover the position (i.e. vertex index) corresponding to the maximum measured value.

1.1 The scenario

For several years autonomous robots have been used as scouts instead of humans in all those occasions when the use of a machine is advisable, due to hard or even dangerous conditions. A few examples: rescue robots looking for survivors after earthquakes or other disasters, deminers or analyzers of packages suspected to contain explosives, data loggers in contaminated environments, to explore seabeds, to clean surfaces in hard positions such as windows of skyscrapers, gutters, swimming pools; they are also widely used in space exploration (fig. 1). Anyone of these situations can be used as a subject for a preliminary review by the students to better contextualize this experience.

Figure 1. Autonomous robots used in hard conditions (source: robotland, zdnet, army-technology, whoi, robots-and-androids, walmart, robotliving, wired)

Our experience is much simpler: to make the robot follow the sides of a regular polygon of n sides and to collect some data, namely the color code of a piece of paper or tape put on each vertex (fig. 2 in the case of a hexagon), reporting the maximum value.

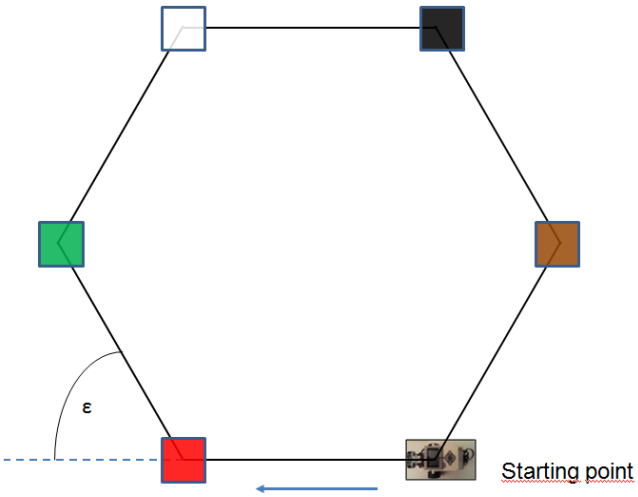


Figure 2. The layout for a hexagon

1.2 Connections with subjects

1. Historical: some science fiction movies imagine humans and machine to be able to take a step back in time. So imagine to have your robot to navigate within an old roman city like Thamugadi (Timgad, Algeria, http://whc.unesco.org/en/list/194). Its regular division in orthogonal streets permits to navigate in any place of the city with straight-line motions and sometimes spinning at intersections (fig. 3). Try to imagine how to code a map of such a regular organization, and to solve the problem to reach a place from another one avoiding possible obstacles.

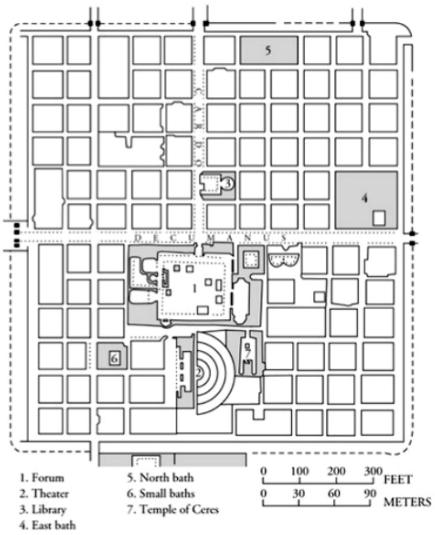


Figure 3. Map of Timgad
(source: http://blog.archaeology.institute/)

2. Visual arts: Colors are coded in different ways. Some code systems are: RGB (Red-Green-Blue), CMYK (Cyan, Magenta, Yellow and Black), HSV (hue, saturation, value), HSL

(hue, saturation, lightness). Study these representations and try to make a comparison in the framework of computer graphics. Refer also to the Goethe's color theory (fig. 4) (see for example https://www.brainpickings.org/2012/08/17/goethe-theory-of-colours/).

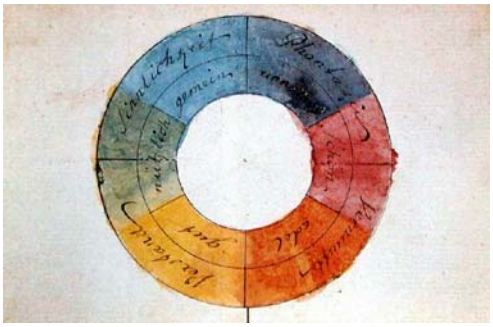


Figure 4. Goethe's symmetric color wheel
(source: http://blog.yovisto.com/johann-wolfgang-von-goethe-and-his-theory-of-colours/

3. Geometry & Philosophy: the Greek era reserved a great importance to the magic of geometry (though this subject was investigated also by other ancient peoples lived before). Try to discover the relationship between polygons and the Greek philosophy.

4. Geometry & History: Frederick II, the famous and powerful Holy Roman Emperor, is the origin of the building of "Castel del Monte" (Andria, Apulia, Italy), which seems a big fortress but several elements show that it was designed for different purposes. It has a double octagonal footprint and presents other geometrical properties of great interest. After reproducing its map on a big sheet, one challenge would be to make the robot follow alternatively one side on the external octagon and one on the internal octagon, 'following' the joining walls. (fig. 5)

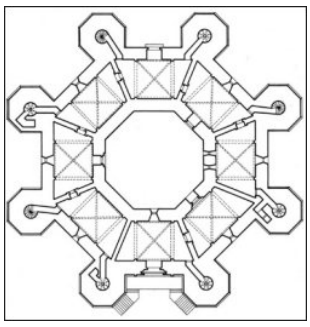


Figure 5. Map of Castel del Monte

Chapter 2: **Pedagogical objectives (O1.2)**

2.1 General objectives

- To provide students with a stepwise approach for a step by step acquisition of technical skills in using robotic technologies (hardware and software) building on existing knowledge and skills.
- To offer the robotics benefits for all children, especially those at risk of school failure or early school leaving.
- To engage students in STEM related subjects through interaction with the robotics technologies.
- To support self-directed action allowing learners to learn independently.
- To engage students in robotic constructions and problem solving through an interdisciplinary scenario that reflects aspects of real-life problems and situations.
- To align the robotics project to learners' needs and interests through tasks that derive from the initial activity but introduce new levels of complexity and difficulty.

2.2 Specific objectives

More specifically, upon successful implementation of the activities described in this curriculum students will achieve the following objectives:

- Learn to mount sensors on the tribot
- Create the mock-up, the environment in which the robot will operate based on the scenario
- Understand through trial and error experimentations the 3 different ways of making the robot turn left or right
- Use the Move Steering command to spin the robot for a certain angle in the context of moving on the lines of a hexagon
- Link trial and error experimentations to the mathematical solution underpinning the given scenario
- Implement and execute the "Turtle total trip theorem" introduced by S.Papert
- Link the rotation of the motor to the turning angle of the robot based on experimentations
- Going deeper into the mathematical reasoning underpinning the aforementioned link
- Understand the use of the light/color sensor in the context of the given scenario

- Understanding what a variable is and using variables for keeping the values provided by the color/light sensor

- Comparing the value of the variables and identifying the maximum value through algorithmic solutions

Chapter 3: **Suggestions for learning methodologies (O1.3)**

The methodology introduced in the previous curriculum is once again implemented here. However, it is tailored to the scenario introduced in the current curriculum. As special worksheet has been designed as a reference and supporting tool. The students are encouraged to work in groups. The teacher acts as a scaffolder and facilitator of the learning process. He/She provides feedback without revealing solutions and probing students through key questions to overcome emerging problems and difficulties. The activity starts with the delivery of the scenario to the students. The teacher in an easy-to-grasp way elaborates on the scenario. The students are encouraged to discuss the given scenario in groups and to form a general methodology for dealing with the problem. The students are then called to create the mock-up, the environment into which the robot will operate based on the scenario of the activity. The activity progresses with the teacher explaining the role of the light/color sensor and showing to the students how to mount it on the tribot.

The students are then encouraged to work with the tribot following the worksheet. They are called to explore through trial and error ways of making the robot turn left or right. The teacher will support the students during this process and will summarize students' findings making the necessary contributions if needed in the end. The students are then experiment with the use of the "Move Steering" command to spin the robot for an angle of certain degrees while this is moving on the lines of the hexagon. The teacher encourages the students to discover the mathematical solution underpinning the given scenario based on the results of the experimentation/trial and error.

In the context of the activity, the students are also called to implement and execute the Turtle total trip theorem introduced by S.Papert. The teacher can gradually challenge students thinking by raising questions related to the application of the theorem on a triangle, a rectangle and finally on a hexagon following the scenario of the activity. Once again, the students are encouraged to playful explore the relationship between the rotation of the motor and the turning angle of the robot based on experimentations. The teacher facilitates their explorations and experimentations. In an attempt to link the exploration to the mathematical reasoning the teacher may require them to describe this relationship using mathematic knowledge and rules.

For the completion of the activity the students are also encouraged to explore the use of the light/color sensor as well as the role of variables for keeping the values provided by the color/light sensor. The teacher discretely supports the students during this process. He/She then encourages them to think of ways to compare the values of variables in order to identify the maximum value. The solutions should then be translated into algorithmic solutions.

A. The role of the students

Students first discuss the problem through a free dialogue in their group and after that they devise an action plan to solve it. They work in groups following the worksheet and the discrete feedback they receive from the teacher. Students may extend their work to variants suggested by the teacher or devised by students themselves. First, they find solutions making trial and error experimentations. In the end they are supported to find a mathematical solution to the same problem. The final solutions of the groups are presented in the class, are discussed and evaluated with students reflecting with critical mind on their work, expressing their views and recording their experiences in a diary or questionnaire.

B. The role of the teacher

The teacher in this constructivist theoretical framework like that described above does not function as an intellectual "authority" that transfers ready knowledge to students but rather acts as an

organizer, coordinator and facilitator of learning for students. S/he organizes the learning environment, raises the questions / problems to be solved through a worksheet, offers hardware and software necessary for students' work, discreetly helps where and when necessary, encourages students to work with creativity, imagination and independence and finally organizes the evaluation of the activity in collaboration with students.

Chapter 4: **Technical guidelines (O1.4)**

4.1 Building instructions

With respect to the structures of the robot used in the 'Roborail' and 'Go to park' experiences, we have to add a light sensor so that it points its sensitive element towards the floor (fig. 6). The actual position of the sensor is not so critic but it obviously influences how to put colored markers on the floor to be reliably sensed by the sensor during the motion.

For the easiest possible layout where to make the robot move, you can start on an empty table, make the robot 'draw' the polygon and to sign the position of the vertices where to put the colored pieces. Even more simply you can ask the students to put by hand the current piece of colored paper under the sensor when the robot stops for a while at a vertex.

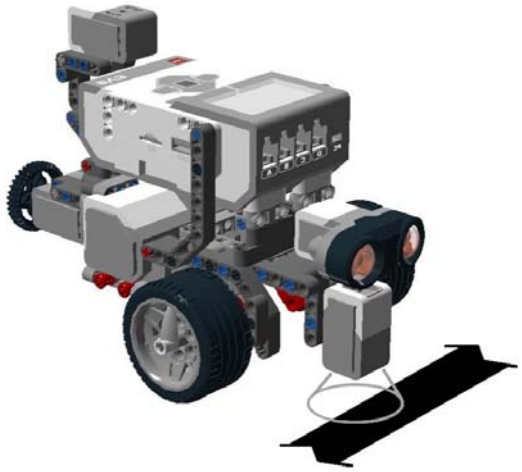


Figure 6. The light sensor mounted (source: legoengineering.com)

4.2 Illustrative solution

Let's start going back to the Roborail experience. In that case for a straight-line motion we suggested to use the `Move Steering` command, leaving the steering parameter to 0, so that the two wheels have the same angular speed and robot does not turn (unless the usual imprecision). As we saw, the power parameter indirectly set such an angular speed. This is transformed into a robot straight-line speed through the wheels having a certain radius.

If we put a value of steering different than 0 in the `Move Steering` command, either positive or negative, we note that the robot turns because the two wheels do not rotate at the same angular speed anymore. More precisely, the slower wheel seems to draw an arc of circle with a radius smaller than the one of the arc drawn by the faster wheel, thus the first becomes the internal wheel during the turn and the other one the external.

Fig. 7 shows the geometric model. A complete discussion on this model is beyond the scope of these notes, but it can be found in this document:
http://www.terecop.eu/downloads/chapter_2.5.pdf

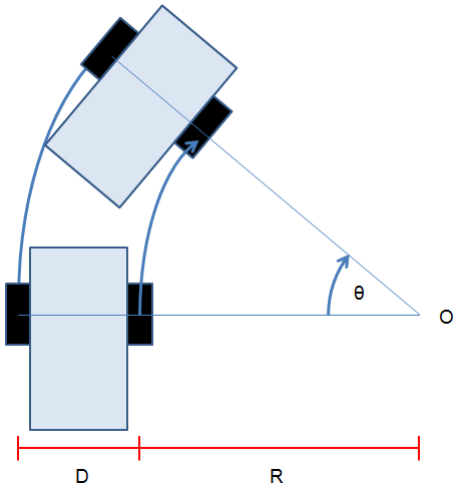


Figure 7. Geometric model of a generic turn

The figure shows that, in principle, the two wheels draw arcs of circle having different radius but with a common centre of curvature O. We can see experimentally that, setting increasing values for the steering parameter, we obtain arcs with smaller and smaller radius until a moment, with the parameter equal to 50, when the internal wheel is kept stopped and 'its' radius is 0, while the radius for the external wheel is D, the distance between the two wheels. If we increase the parameter in the range 50..100, the internal wheel is rotated in the opposite direction with respect to the external wheel: the center of rotation is within the space between the two wheels and the robot rotates around itself (a motion called 'spin' or 'pivot'). If we make the same experiment with negative values of steering, the effect is similar but the robot turns left instead of right. A more detailed explanation of this model is given in the appendix of this document.

When the parameter is 100, the radius of curvature of both wheel is D/2 and therefore the centre of rotation is exactly in the middle of the two wheels which rotate with the same but opposite speed (fig. 8).

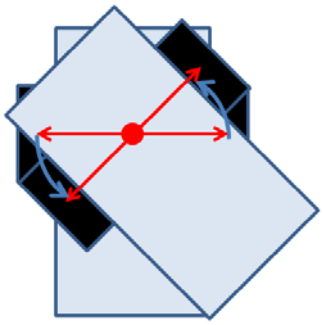


Figure 8. Spinning

In the special case of steering=100 (or -100), it can be shown that the relationship between the angle of spinning of the robot θ and the angle of a wheel α is given by:

$$\alpha = \theta \cdot D / (2 \cdot r)$$

with r the (equal) radius of the wheels. This result is meaningful because we need to turn around a fixed point relative to the robot, maintaining its natural symmetry. This is what happened for the traditional Logo turtle and it is still the case for sprites in the Scratch (https://scratch.mit.edu/) and Snap! (http://snap.berkeley.edu/) environments.

Now we can address the main task. It is easy to recognize again that we have to execute a sequence of actions that must be repeated for the number n of sides: one move straight, one reading with the light sensor, one spin. The straight-line motion was already discussed in the Roborail experience, so we discuss now the other two actions.

Our task requires to collect the values read by the light sensor on each vertex. Because we must find the maximum among these values, it is not necessary to store all the read values, it is easier to update a temporary maximum, and the index of the corresponding vertex, during the visit of the successive vertices. For this purpose, we can use some variables: *current* to store the current read value, *index* that represents the current vertex index (it is provided by the *loop* block from 0 to n-1), *max* and *position* that store the temporary maximum and corresponding index.

For reading the value from the color sensor, we use one of the commands present in the *Sensor* category called `Color Sensor` (fig. 9).

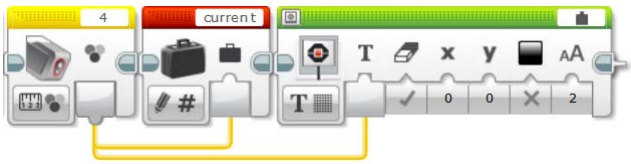
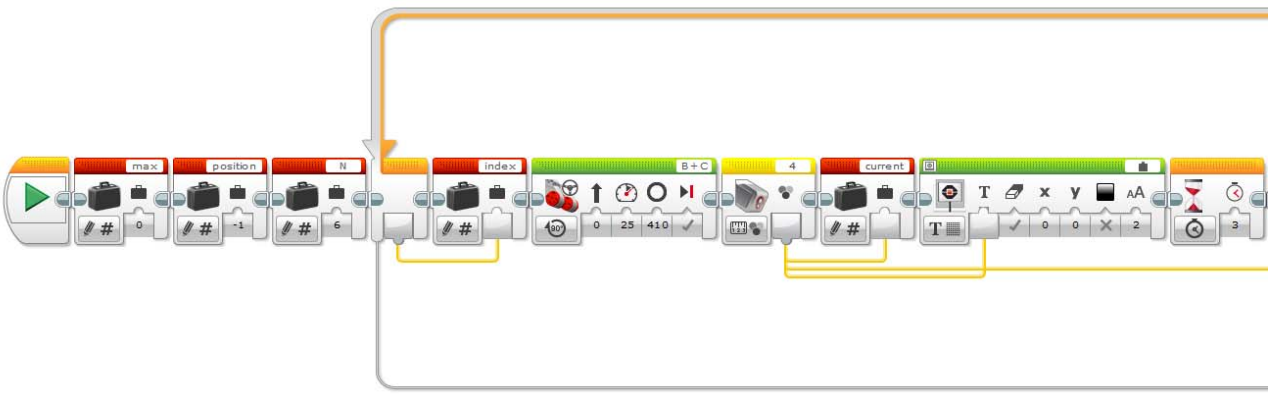


Figure 9. Reading from the Color *sensor and displaying*

The read value is transferred to following blocks through a data wire. We store the read value into the *current* variable with a write block and we also suggest to display the value on the brick's screen with the `Display` command (this is advisable to control whether the color code is the expected one). Take notice of the parameters necessary in this case as shown in fig. 9: the text is *wired* that is provided through a data wire, and the conversion from a number, the read value, and the text to be displayed is implicit.

At each vertex the robot must spin for a number of degrees which is easy to calculate. The "Turtle total trip theorem", presented by S. Papert in his famous book "Mindstorms: Children, Computers and Powerful Ideas", says: "If a Turtle takes a trip around the boundary of any area and ends up in the state in which it started, then the sum of all turns will be 360 degrees". So, due to regularity, the amount of degrees to turn on each vertex of a n-sided polygon is always the same on each of them (the angle ε in fig. 2) and the theorem above implies this amount is 360/n. As already explained, spinning is obtained by using a `Move Steering` command with the Steering parameter set to 100 or -100 depending on the direction of rotation.

Now we have all the elements for the complete program shown in fig. 10.



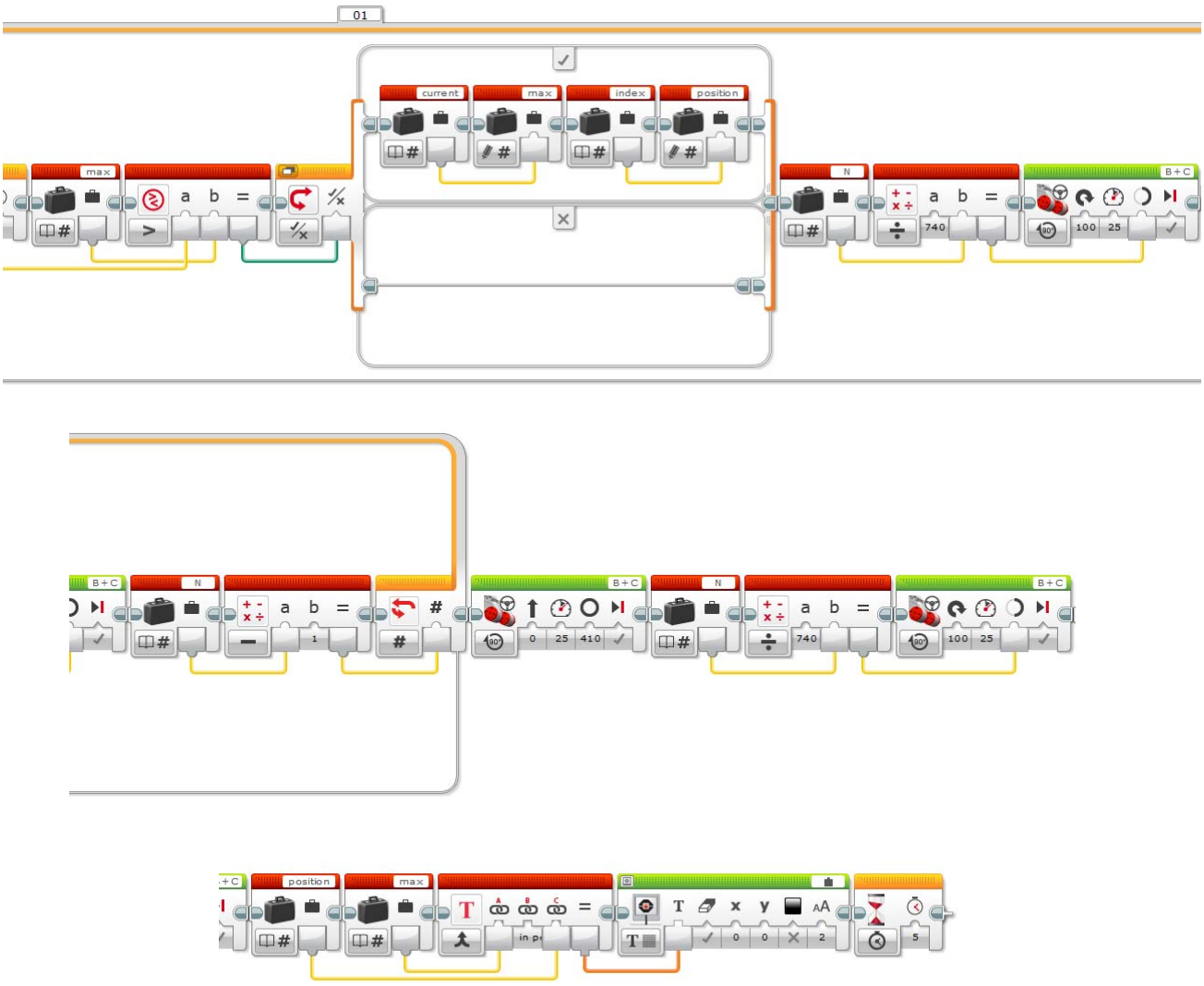


Figure 10. The complete program

We assume that the starting vertex represents a sort of docking station for the robot and no reading with the sensor is requested in that position. For this reason, though the variable N is initialized with the number of sides, the loop is executed N-1 times which is the number of readings to be collected, starting from the vertex following the docking station. In the loop the operations to be performed orderly are:

- Straight motion for 20 cm (r=2.8 cm), it means: degrees = (180/π) * space/r = 57.3 * 20 / 2.8 ≈ 410;
- Read the color, save and display the color code;
- Wait 3 seconds;
- Possibly update *max*: the current max is compared with the current read value, if the last is greater, update *max* with this value and *position* with the current *index*; the conditional update is done within a `Switch` command (for a more detailed description see the next curriculum);
- Spin for 360/n degrees (D=11.5): the angle for which both wheels must be rotated is $\alpha = \theta \cdot D / (2 \cdot r) = (360/n) \cdot 11.5 / (2 \cdot 2.8) \approx 740/n$;

- Use N-1 as the repetition count;
- Execute the final straight motion and spin to return to the initial state at the docking station without reading;
- Display position and value of the maximum and leave it displayed for 5 seconds.

4.3 Implementation suggestions

With this experience you are asking the students to have a deeper insight on the robot behavior. You are also completing the introduction of the most relevant commands in EV3-G, so that they have all the basic tools to address problems of increasing complexity.

It is not necessary to show to the students all the theory which underlies the robot behavior: some formulas and direct experience are enough for understanding the spin action of the robot also quantitatively. But the experience suggests several arguments of deepening, about geometry, trigonometry, color representation, data acquisition and analysis.

At this point the students should also have a full understanding of the usefulness of variables.

4.4 Extensions and variants

4.4.1 Variant a: Go to max for the extraction [easy]

Using the position of the maximum stored in the *position* variable, make the robot run through again the polygon sides to reach the vertex corresponding to the maximum value. You have essentially to add a new loop, similar to the first one, containing the straight motion and spin and repeated a number of times given by *position*. But take notice that in a loop the internal index starts from 0.

4.4.2 Variant b: The turtle [easy]

Define four user blocks, *forward*, *backward*, *spinright*, *spinleft*, representing respectively a straight-line motion of *s* cm with power *p* forward, the same backward, spin for *a* degrees with power p clockwise, the same counterclockwise. *s*, *p* and *a* are command parameters.

Fig. 11 shows the *forward* user block which is analogue to the *nextslot* of the previous experience.

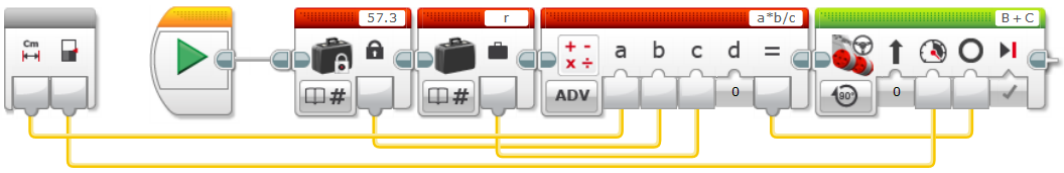


Figure 11. The forward *user block*

The *backward* user block is similar but with inverted motion direction. Fig. 12 shows the *spinright* user block (*spinleft* is similar, only invert steering).

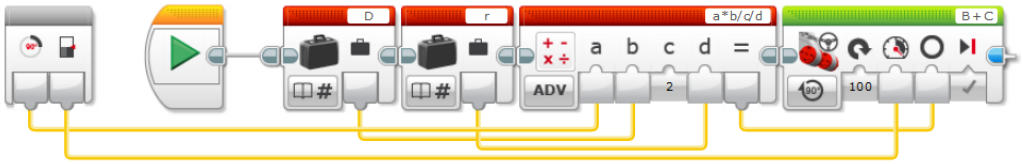


Figure 12. The spinright *user block*

These blocks refer to variable D and r as general parameters set outside.

4.4.3 Variant c: Go direct for the extraction [medium/difficult]

Differently from the previous extension, the robot is requested to spin at the docking station for a suitable angle in order to point directly to the vertex with the maximum value. The difficulty of this extension is essentially due to the comprehension of the geometrical layout because the robot has only to spin for a certain angle and move straight for a certain distance, provided you have calculated that angle and that distance according to the vertex to be reached.

In general terms, let's first recall that a regular n-sides polygon, each side of length L, is always inscribable in a circle of radius R. (fig. 13):

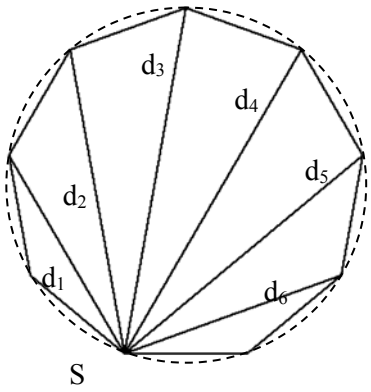


Figure 13. A nonagon and its six (internal) diagonals from S

From the chord theorem it holds (angles in degrees):

$L = 2 \cdot R \cdot \sin(360 / (2 \cdot n)) = 2 \cdot R \cdot \sin(180 / n) \quad \Rightarrow \quad R = L / (2 \cdot \sin(180/n))$

The sum Sa of internal angles is:

$Sa = 180 \cdot (n-2)$

thus each internal angle is wide $180 \cdot (n-2) / n$ degrees. It is easy to show that each one of the (n-2) angles which are in between two successive diagonals from the same vertex, or one diagonal and the adjacent side, has the same width because these angles intercept arcs, and chords/sides, of the same size. Thus these angles are wide $[180 \cdot (n-2) / n] / (n-2) = 180 / n$.

Let's call d_i the diagonal from the starting vertex to the i-th successive vertices (0..i..n-2, from left to right); for clarity, we include the two sides departing from the chosen starting vertex with index 0 and n-2. It can be shown that the length of each one of them is:

$d_i = 2 \cdot R \cdot \sin((i+1) \cdot 180 / n) = L \cdot \sin((i+1) \cdot 180 / n) / \sin(180 / n)$

Because sin(α)=sin(180-α) we have also:

$d_i = L \cdot \sin((i+1) \cdot 180 / n) / \sin(180 / n) = L \cdot \sin(180 - ((i+1) \cdot 180 / n)) / \sin(180 / n) =$

$= L \cdot \sin((n-i-1) \cdot 180 / n)) / \sin(180 / n) = d_{n-i-2}$

that is diagonals are coupled, one on the left, one of the right with the same size (one is alone if n is even).

For example, for a regular nonagon, the internal angle is wide 140 degrees and we have:

d_0	L	d_1	$L \cdot \sin 40/\sin 20$
d_2	$L \cdot \sin 60/\sin 20$	d_3	$L \cdot \sin 80/\sin 20$
d_4	$L \cdot \sin 100/\sin 20 = d_3$	d_5	$L \cdot \sin 120/\sin 20 = d_2$
d_6	$L \cdot \sin 140/\sin 20 = d_1$	d_7	$L \cdot \sin 160/\sin 20 = L = d_0$

Two very interesting cases are with n=5 and n=6. For a pentagon, the two internal diagonal, excluding the sides, have both a size equal to:

$d = d_1 = d_2 = L \cdot \sin 72 / \sin 36 \quad \Rightarrow \quad d/L = \sin 72 / \sin 36 = 1.61803$

which is the famous *golden ratio* (or *section*).

For a hexagon R=L and results:

$d_1 = d_3 = L \cdot \sin 60 / \sin 30 = L \cdot \sqrt{3} \qquad d_2 = L \cdot \sin 90 / \sin 30 = 2 \cdot L = = 2 \cdot R$ (a diameter)

Concluding, say i the index of the vertex corresponding to read max value, after the first scouting round, the robot must spin $i \cdot 180 / n$ degrees and move straight for the length d_i calculated with the formula above.

4.4.4 Variant d: Go there [medium]

Having a N-sided regular polygon with N parameter, make the robot move and stop at the i-th following vertex with i coded by a color showed to the robot through the color sensor. The variant becomes of the same difficulty level of c) if you ask to go directly to the target vertex.

4.4.5 Variant e: Optimize [difficult]

Similar to the c) variant but now the target position to be reached must be calculated as the maximum of the ratio gas quality/straight-line distance. This means that all the readings during the scouting travel must be stored into an array and, at the end of this phase, the maximum ratio must be calculated elaborating all the values in the array.

4.4.6 Variant f: Draw polygon [medium]

The robot moves on a straight segment from a certain point until it reaches a marker indicating the end of the segment; the size of the segment, evaluated by the robot, can be considered the radius of a circumcircle or an incircle (apothem) of a N-edge regular polygon; the robot has to 'draw' this polygon so that it is shown that the starting point is the centre of that circle. For calculations see the theory described in the variant c).

4.4.7 Variant g: The blind escape [difficult]

Using the Bluetooth connection, make the robot spin regularly and give the distance of all the objects in front of the robot within a certain angle. These objects must be avoided by the robot: the student receives this series of measures on the PC, draw manually a map of the measured scenario

(or possibly using a computer tool) and decides the angle to turn in order to make the robot move ahead beyond any obstacle.

Chapter 5: **Evaluation tools (O1.5)**

Use the rubric below to evaluate your students' achievement in each specific objective of this curriculum.

Name of student (or group of students): ……

upon completion of the activities described in this curriculum students achieved the following objectives	**Evaluation score** **0 = not attempted** **1 = attempted without success** **2 = partial success** **3 = completed with teacher's help** **4 = completed without teacher's help**
Properly mounted and used a color sensor both for detecting colors and measuring reflected light	
Prepared carefully the scenario using colored tapes or pieces of paper	
Realized the spinning motion linking the rotation of the motor to the turning angle of the robot	
Discovered and properly used the "display" command	
Discovered and properly used the "move steering" command to make the robot spin	
Instructed the robot to travel through a regular polygon of a given number of sides	
Instructed the robot to travel through a regular polygon of a generic number of sides	
Properly used variables for keeping the values provided by the color sensor	
Instructed the robot to find the 'maximum value' of the color code	
Properly defined their own user blocks	
Designed a mathematics-based solution for making the robot travel through a polygon	

Chapter 6: **Appendix**

To have a deeper insight in the geometrical model, let's recall the main conclusions. You can observe that the two wheels draw arcs of circle with different radius but a common center of curvature O. Say R the distance between O and the point of support of the internal wheel, and D, like before, the distance between the two points of support, and θ the angle of turn of the robot, it holds:

$R = D \cdot \omega_i / (\omega_e - \omega_i)$ $\theta = \alpha_i \cdot r / R = \alpha_e \cdot r / (R+D)$

with ω_i, α_i and ω_e, α_e angular speed and rotation of, respectively, the internal and the external wheel, measured in corresponding units (you must use either radians and radians/s, or degrees and degrees/s).

We have experimentally checked that, as a first approximation, there is a direct proportionality between the $_{value}$ of the power parameter and the wheel angular speed, and consequently the robot speed. The diagram of fig. 14, plotting angular speed ω_g versus power P, with ω_g measured in degrees/s and P is a pure number (0..100), shows that at least in the range of power 10-80 it holds:

$\omega_g = K_{\omega gP} \cdot P$

Our experimentation gave for KωgP an average value of about 10.1 degrees/s. For the straight-line speed v we have:

$v = \omega \cdot r = (\omega \text{ in radians}) = \omega_g \cdot (\pi/180) \cdot r = K_{\omega gP} \cdot (\pi/180) \cdot r \cdot P = K_{\omega P} \cdot r \cdot P$

where v is measured in cm/s if r is given in cm, and then:

$K_{\omega P} = K_{\omega gP} \cdot (\pi/180) \approx 0.176$ 1/s

For example, with r=2.8 cm, the speed with P=80 is about 39.5 cm/s.

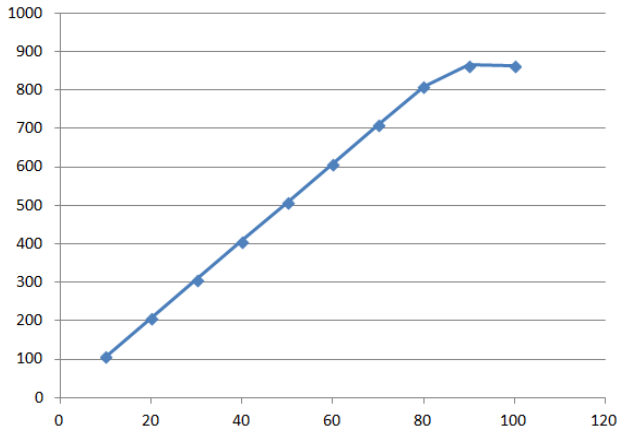


Figure 14. Angular speed in degrees/s versus power

We evaluated experimentally the relationship between the steering parameter, which can vary from -100 and 100, and the angular speeds. Positive values of the steering factor induce a right turn,

negative values a left one. In fig. 15 the two angular speeds at the intermediate power P=40 are plotted versus a steering factor varying from 0 to 100. You can observe that the external wheel maintains substantially unaltered its speed while the internal wheel decreases almost linearly its angular speed until it becomes 0 with a steering factor of 50. Therefore, with this setting we obtain the type of swing turn that is used in the 'Go to park' experience. A bit surprisingly at a first glance, if you continue to increase the steering factor, the internal wheel rotates backward until it rotates at the same but opposite angular speed of the external wheel. Evaluating for the plotted values of angular speed the corresponding ratio R/D = ω_i / (ω_e – ω_i) we obtain the curve plotted in fig. 16, which is, by the way, approximately a hyperbole fitted by the function R/D = (-steering+50)/steering.

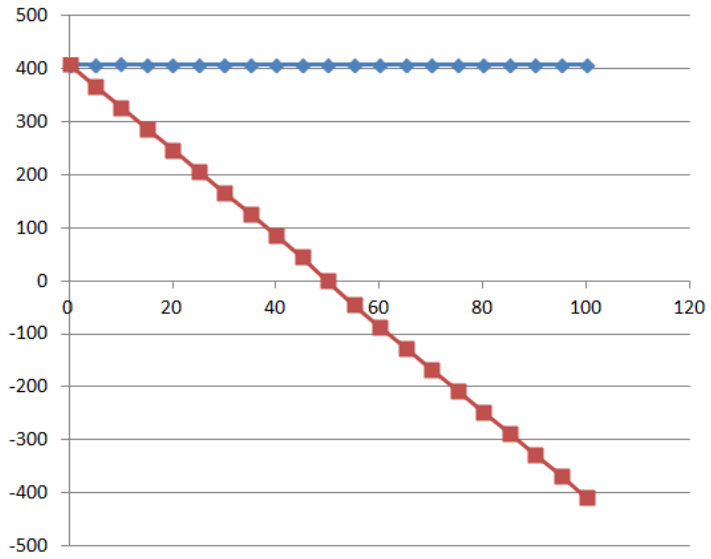


Figure 15. The two angular speed versus steering (P=40)

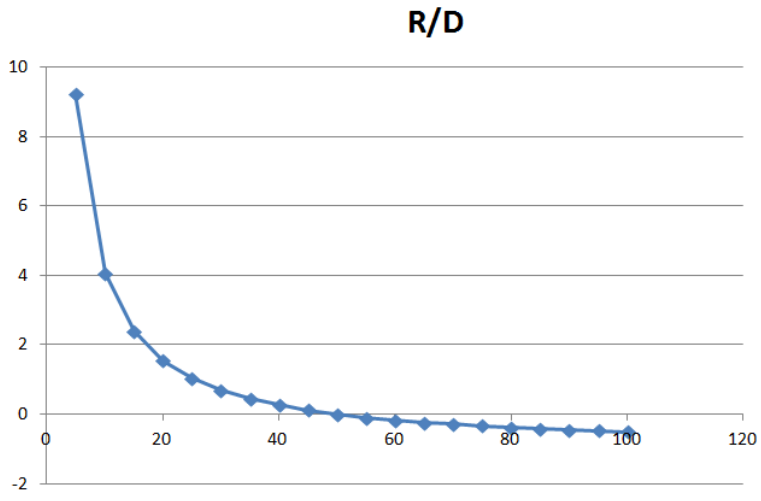


Figure 16. The ratio R/D versus steering (R/D = [-steering+50]/steering)

With steering>50 the centre of curvature is neither outside the robot, nor under one wheel but between the two wheels. When the steering factor is 100, because ωi = -ωe, it holds:

$R = - D/2$ $\theta = \alpha \cdot 2 \cdot r / D$ $\alpha = \theta \cdot D / (2 \cdot r)$