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 4: Let's play and dance

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 Where other published and unpublished source materials have been used, these have been acknowledged.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. 4, entitled ‘Let’s play and dance’ 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. This Curriculum is dedicated to introduce some new features: the line following principle, useful in several other situations; how to make the robot move while it concurrently reproduces sound or tunes; how to add random behaviors in the robot actions/decisions.

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

(source: theroboticist.org)

Our (dressed) robot becomes a dancer: it enters the stage, possibly following a guide line, and then it performs a small choreography while it plays a tune. Dancing and playing are organized as concurrent actions.

1.1 The scenario

This is an experience for which you can ask the students to prepare richer or less rich scenarios where to make the robot dance. We suggest to think to a hypothetical stage, the stage can be reachable through a leading line (imagine something like a series of pale lights in the darkness). On the stage the robot makes a sort of dance following the music the robot itself plays (fig. 1).

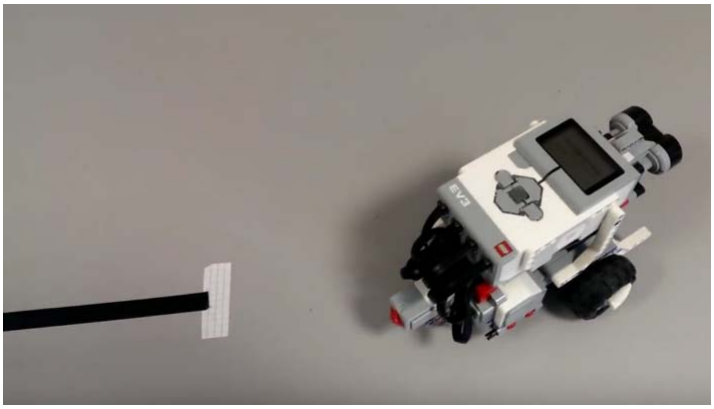


Figure 1. The 'stage'

1.2 Connections with subjects

This curriculum has relationships with music, theatre and other subjects.

1. History, Music, Mathematics and Physics: since the Greek Era, music production has to face the problem called *temperament* which, in extreme synthesis, is the necessity to accommodate (tuning) the frequency of the sounds produced by an instrument with fixed sounds, like an organ, a piano or in general keyboard-based instruments, so that the perception is of a good harmony with any tonality. Historically 4 types of tuning were applied in different ages: Pythagorean tuning, just-intonation, mean-tone and equal temperament. The problem arises from acoustic physics: we show it through an example.

We are used to consider a spectrum of 12 tones per octave. Modern tuning is based on a reference frequency for A4, the *La* note of the forth octave, established in 440Hz. The frequencies of A5, A6, A7, A8 are easily calculated because we know that:

fA(i+1) = 2 * fAi

Therefore fA5 = 880, fA6 = 1760 etc. But we also know that any periodic signal, sound included, may be considered as the sum of an ideally not limited number of sinusoids which can have different amplitude and phase but whose frequency is an integer multiple of the frequency of the signal; they are called *harmonics*. Therefore, when a non sinusoidal sound is produced we can hear several harmonics but only a few of them correspond to the same note: this is a more evident experience if you hear to the echo produced by an instrument played, for example, in a big church. Which notes are also included in the emission? Let's continue the example with the following table showing the harmonics of a note A4, measured in Hz (Hertz):

i	1	2	3	4	5	6	7	8	9	10
f	440	880	1320	1760	2200	2640	3080	3520	3960	4400
Name	A4	A5	E6	A6	C#7	E7	F#7	A7	B7	C#8
i	11	12	13	14	15	16	17	18	19	20
f	4840	5280	5720	6160	6600	7040	7480	7920	8360	8800
Name	D8	E8	F8	F#8	G#8	A8	A#8	B8	C9	C#9

When you play a chord of A major, you play the tonic A, the third C#, the fifth E and the successive A: the perception is of harmony because the table above shows that C# and E are among the first, and more relevant in amplitude, harmonics of A.

Now, consider to list the harmonics of a note E4. Its frequency should be two octaves under the frequency of E6, a harmonic that we found in the A4 spectrum with frequency 1320Hz. We obtain the following table:

i	1	2	3	4	5	6	7	8	9	10
f	330	660	990	1320	1650	1980	2310	2640	2970	3300
Name	E4	E5	B5	E6	G#6	B6	C#7	E7	F#7	G#7
i	11	12	13	14	15	16	17	18	19	20
f	3630	3960	4290	4620	4950	5280	5610	5940	6270	6600
Name	A7	B7	C8	C#8	D#8	E8	F8	F#8	G#	G#8

So the harmonics of the second group have frequencies not exactly correspondent, when the sound is apparently the same, to the harmonics of the first group. This means, for

example, that C#7 'included' in the harmonics of A4 is different from the same note C#7 expected as an harmonic of E4. A chord of E major including C#7 with the frequency of the first table could be perceived a bit disharmonic by a trained ear.

To reduce this effect note frequencies are tuned to obtain an acceptable compromise. The 12-tone equal temperament establishes a total of 12 successive semitones in an octave so that the ratio of frequencies of two successive notes is constant. This constant K can be easily calculated:

$A4=440 \quad A\#4=440*K \quad B4=440*K^2 \quad C5=440*K^3 \quad \ldots$
$A5=440*K^{12} = 880$

$K = \sqrt[12]{(880/440)} = \sqrt[12]{2}$

Using this constant, the notes in the A4-A5 octave so tuned result as:

f	440.00	466.16	493.88	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99	830.61	880.00
Name	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5

2. There are two interesting examples of automatic theatre, one coming from the ancient Greece and one more recent.

   Heron of Alexandria (about 10-70 AD) was a Greek mathematician, engineer and an extraordinary inventor of automatic machines, centuries before Leonardo da Vinci. Among these he proposed an automatic theatre about the myth of Dionysos (see a 3D reconstruction in https://www.youtube.com/watch?v=5LBlusUD3Kg).

   In the first half of the 20th century the Futurist movement suggested several revolutions in literature and arts, and one of these regarded the theatre. In opposition of the tradition, it suggested that modern theatre should have been more synthetic, based on parody and with a scenography more dynamic, abstract and with a huge use of electric light. Around these ideas Fortunato Depero, an Italian painter and sculptor, very active in advertising production, together with the Swiss poet Gilbert Clavel, created the "Plastic ballets" where he imagined to substitute actors and dancers with puppets with rigid movements, a sort of automata which will have then filled his pictures and posters (fig. 2).

a) *b)* *c)*

Figure 2. Fortunato Depero – a)Automi, prospettiva dinamica figurata (Automata, Dynamic Perspective with Figures), 1917 – b) Gli Automi (The Automata), 1945 – c) I miei balli plastici (My plastic ballets), 1918

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 STEAM related subjects through interaction with the robotics technologies and craft making.
- 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:

- Creatively decorate the robot for the needs of the given scenario: this means that the students will be engaged in craft design and making
- Create a mock up, the environment within which the robot will operate based on the scenario
- Learn to mount the light sensor on the tribot for detecting colored lines drawn on the floor
- Understand the way the light sensor works and to use it in the context of the given scenario
- Understand the way the programming concept of "switch" operates and how to apply it when building an algorithmic solution that instructs the robot to follow a random line
- Understand the way the programming concept of "loop" operates and how to apply it when building an algorithmic solution that instructs the robot to follow a random line
- Make the robot follow a random line on the floor
- Understand the way the "sound" block/command works and to be able to change parameters as needed

- Understand how the “random” block/command for producing numeric and logic values works
- Synchronize the movement of the robot based on the notes played
- Understand the concept of parallel programming
- Creatively synthesize a set of dancing movements in relation to the sound/rhythm played
- Get familiar with musical notes and tones

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

This curriculum follows the ideas underpinning the previous curricula. A 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 introduced through the scenario. This activity differs from the ones already introduced because this time the students are not only called to create the mock-up, the environment into which the robot will operate based on the scenario of the activity, but also to get involved in craft-making and to creatively decorate the robot. 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 robot.

The students are then encouraged to work with the robot following the worksheet. They are called to reflect upon possible ways of making the robot to follow the line. The teacher will support the students during this process and will summarize students' findings making the necessary contributions if needed in the end. With the support of the teacher the students are introduced in the programming concepts of "switch" and "loop". Then they are encouraged to practically apply these concepts in order to make the robot follow a line drawn in the floor.

As the activity progresses the students are exposed to the use of the Sound command/block. The students are offered the time needed to explore these blocks/commands and to use them in the context of the given problem. The teacher acts as a facilitator, offers supports when needed and with prompt questions aims at helping students overcoming any emerging problems.

Then the teacher introduces the block/command for random logic and numeric values. With the support of the teacher the students generate input for the sound block using the random block/command. The teachers also trigger students thinking on how to generate random input for the motor block and encourages the students to discuss in groups how to synchronize the movement of the robot with the notes played. Ideas are collected and with teacher feedback the students start implementing their algorithmic solutions. The teacher moves students to playfully explore a number of alternative combinations of dancing movements and broadcasted notes/rhythms. The students should free their imagination and regulate the dancing behavior of the robot!

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

For the technical part, there is no particular new instruction: mount the color sensor in the same position as in the previous curricula. But according to the artistic flavor of the experience, you can ask the students to enrich the construction, adding pieces which can dress the robot so that it can look like an animal o a human.

4.2 Illustrative solution

This experience must be thought divided in two parts: the first is a line follower, when the robot enters the stage, and the second is the dancing phase. The line following would not be a compulsory preamble but it is a technique useful in many situation and we take the opportunity to introduce a little improved version in this experience.

Line following means a reactive motion of the robot which follows the ideal line that delimits two zones of different color, or of different lightness in the gray scale, drawn on the floor or on the table (fig. 3). A simple solution is to use one light/color sensor, positioned like in the previous experience towards the floor, to discriminate the two colors, or the two different levels of lightness, and conditioning the motion of the robot so that it corrects its direction to be maintained parallel to that ideal separation line.

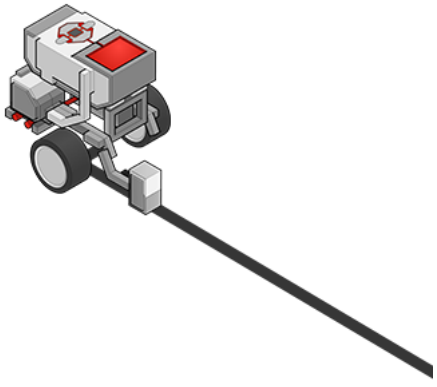


Figure 3. Line follower principle (source: cs2n.org)

The simplest version of this technique requires to distinguish if the reflected light is higher or lower a certain threshold. Accordingly, the program makes the robot turn towards the dark area or towards the light area (fig. 4). The motion of the robot is similar to the walk of a duck. If you want a smoother motion, you can provide an improved version.

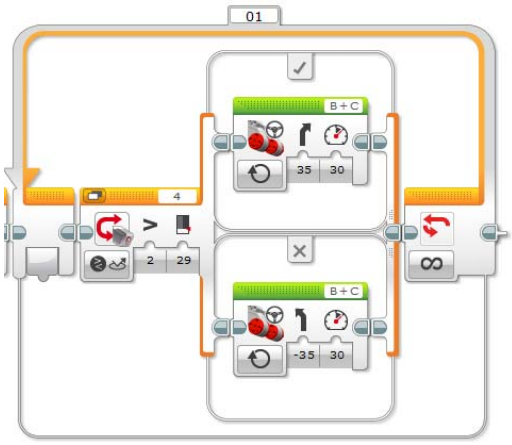


Figure 4. The simplest program

The better solution is to adapt the steering factor to the position of the robot with respect to the reference line. This is possible because the red light emitted by the device is not punctual, it is rather a small bright circle which covers partly the dark, partly the light area (fig. 5).

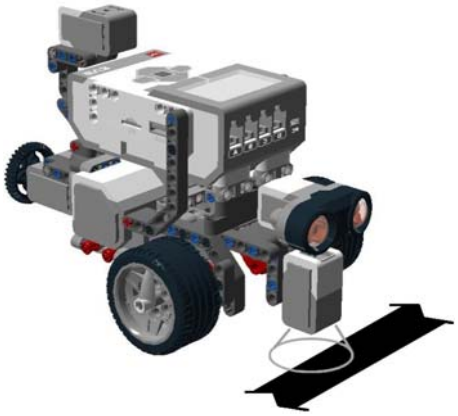


Figure 5. The illuminating light (source: legoengineering.com)

That means that the reflected light, and its measure, varies continuously from the position in which the entire circle of light, or most of it, and particularly the light sensitive component of the sensor, are on the dark area and the position in which they are completely over the light area. Say m and M respectively these two measures (m < M), we want that the steering factor is S (turn right), when the sensor measures M and, for symmetry, -S (turn left) when it measures m; for intermediate values, we impose a proportional (linear) variation. You could also decide to use the calibration method provided for this type of measure (reflected light) in which case what changes is that m=0 and M=100.

steer = f(light) = a • light + b
It must hold:
f(m)=-S and f(M)=S
therefore:
(steer + S) / (2 • S) = (light – m) / (M – m)
Solving:
steer = light • 2 • S / (M – m) – S • (M + m) / (M – m)
With calibration done:
steer = light • S / 50 – S = (light/50 -1) • S

The formula to apply is:

steer = light · 2 · S / (M – m) – S · (M + m) / (M – m)

For example, with S=50, M=30, m = 5, with the simpler solution you would have put (30+5)/2 = 17.5 as the more reasonable threshold; with the improved solution you have:

steer = light · 100 / 25 – 50 · 35 / 25 = light · 4 – 70

The measure is provided as usual through a data wire and transformed by using the Math command, set with its advanced mode: this block alone can make the whole linear calculation of the previous formula, giving the steering factor. It is advisable not to set a great value of power for maintaining smooth the motion (fig. 6).

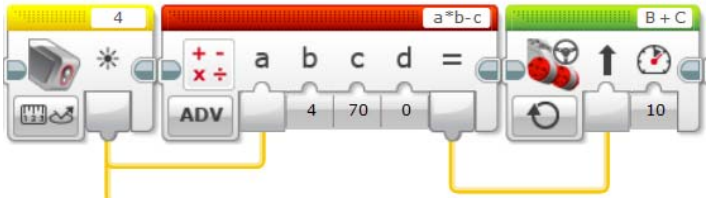


Figure 6. Linear function for steering

Then we insert the segment of code within a loop and we decide to make the robot recognize the end of the leading line putting a piece of white paper at that end: when the measure of the light sensor exceeds a relatively high predetermined threshold, we exit the loop and move a bit on a straight line to 'enter' the centre of the stage (fig. 7).

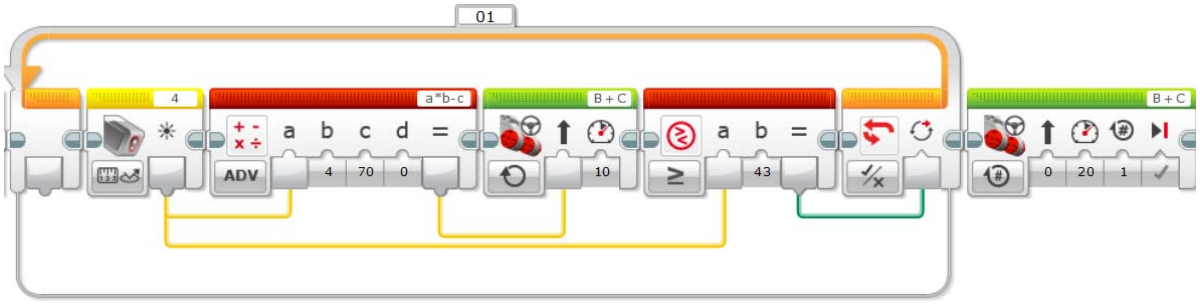


Figure 7. Line following phase

We have now to briefly introduce some details of the Sound command (fig. 8).

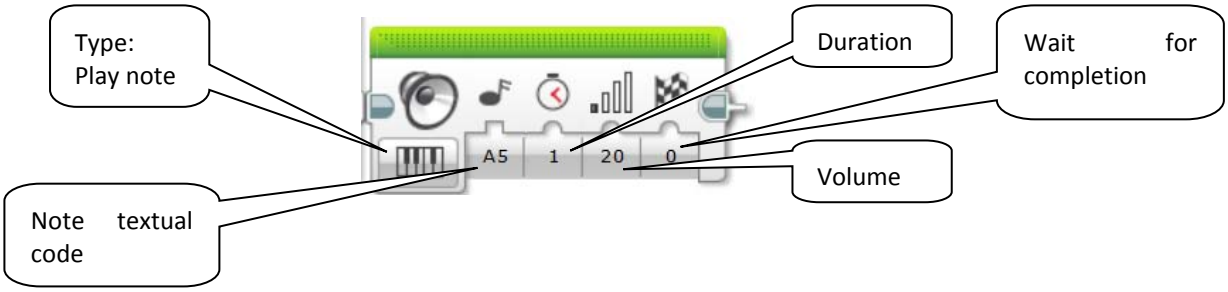


Figure 8. The Sound *command*

As you notice, we use the *Play note* option which is represented as a note to be played on a keyboard and, for this reason, represented by the standard textual code, one letter for the note (A=La, B=Si/Ti, … G=Sol), one optional sharp symbol (#, the flat symbol is not reproducible, substitute it with the equivalent sharp, e.g. F# instead of G♭) and an octave number (4, 5 or 6). The

admitted three octaves are in the ranges C4..B4, C5..B5, C6..B6. For example, B5 is a Si/Ti immediately followed by C6 which is a Do of the following octave.

The first, simplest variant of this experience generates a sequence of random notes together with a synchronized sequence of random little motions. The fundamental block to produce this random behavior is the `Random` command (fig. 9). It can be set to produce numbers or logical values. In the first case every time it is executed it produces an integer value in the given range (-5..7 in the figure), with all the numbers in that range with the same probability to be generated; in the second case one of the two logical values is generated, and the probability of the true value is set as a parameter of the block (40% in the figure).

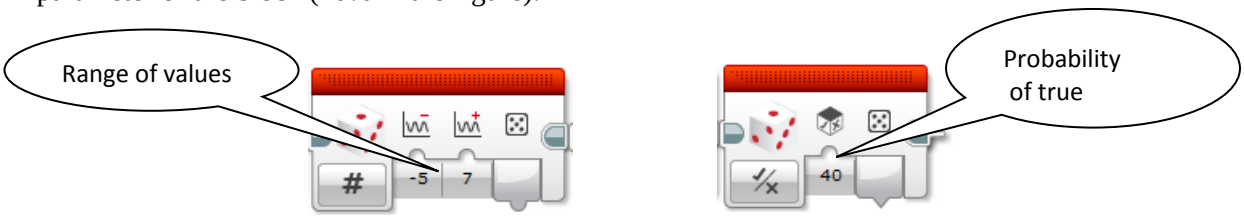


Figure 9. The Random *command, in the numeric and logical modes*

Now we can decide to generate sounds in a certain range, for example between 400 and 1000 Hz, and to use a `Move Steer` command with fix power but a random steering factor between the two extremes -100 and 100. The simple trick to maintain synchronized every sound with the concurrent motion is to define the same duration for both and to put this commands within the same loop. How is it possible to execute commands concurrently? EV3-G provides a specific procedure to define concurrent execution of commands, consisting in dragging a new sequence wire (i.e. the wire connecting chunks of blocks) and inserting each concurrent sequence within one of the parallel wires. The complete program is shown in fig. 10. At each loop cycle the random sound and the random motion have the same duration (1 s). Choose the sound volume as you prefer.

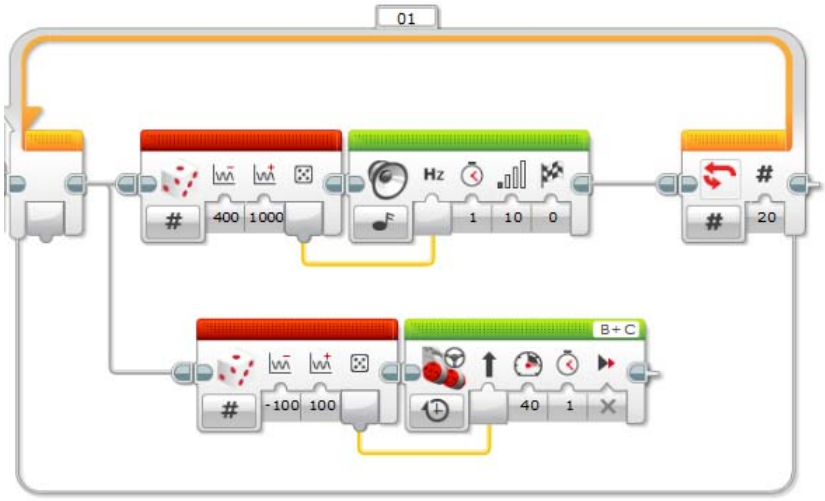


Figure 10. A simple random dance

4.3 Implementation suggestions

As already suggested, in this case the robot can be put in an enriched scenario to simulate a dance stage. Moreover, this curriculum introduces some new interesting commands (Sound, Random) which is worth to deepen in all their possibilities. Also concurrency may introduce some preliminary discussions on parallelism and delicate synchronization issues it can manifest.

The Random block generates integer numbers with uniform distribution of probability. If you want to generate number with fraction, you have to scale the command output: for example, if you set the range between 0 and 100 and then divide the output by 100, you obtain a random generator outputting uniformly distributed values between 0 and 1 with two decimal digits of resolution. If you would have a different distribution of probability, there are general methods to choose the suitable output transformation but these are a bit too complex to be considered here.

4.4 Extensions and variants

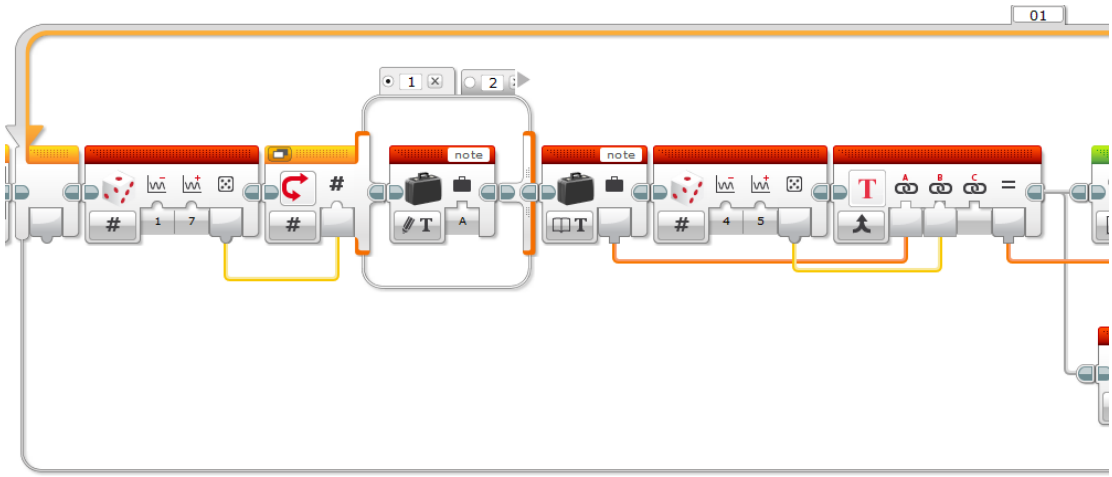
4.4.1 Variant a: More random [easy]

We want to enrich the solution providing the generation of a random note in the range A4..B5 and a more elaborated motion where also the power is not fixed but we want it is -40 (backward) or 40 (forward) with equal probability (fig. 11).

If you observe the Sound block, when you choose the *Play note* option, you notice that the first octave is between C4 and G4 plus A4 and B4, similarly for the second octave (C5..G5, A5..B5). To generate one of 7 letters, it is necessary to transform a number randomly generated into the corresponding letter: this can be done using the *Numeric* form of the Switch block, and orderly associating the numbers 1..7 with the A..G letters, one for every Case value. In fig. 11 the switch is shown with the Tabbed view, so here you see just the case '1' with the letter 'A': consider to have all the other 6 cases with numbers 2..7 and letters 'B'..'G'.

Once produced the letter and the octave number, they are concatenated with the Text block to form a unique string of two characters.

The generation of the power value is given by a binary random generation transformed into one of the two expected values (-40 or 40). Instead of solve the problem with the *Logic* option of a Random block, we suggest to generate one of the two numbers 0 and 1 and to transform the output so that 0 corresponds to -40 and 1 to 40. The formula is simple: say r the random number, the formula is -40+r*80 (notice that the solution to put the range of the Random block (-1, 1) and multiply it for 40 would be incorrect because with that range the generator produces one of three values, -1, 0 and 1, and not only two).



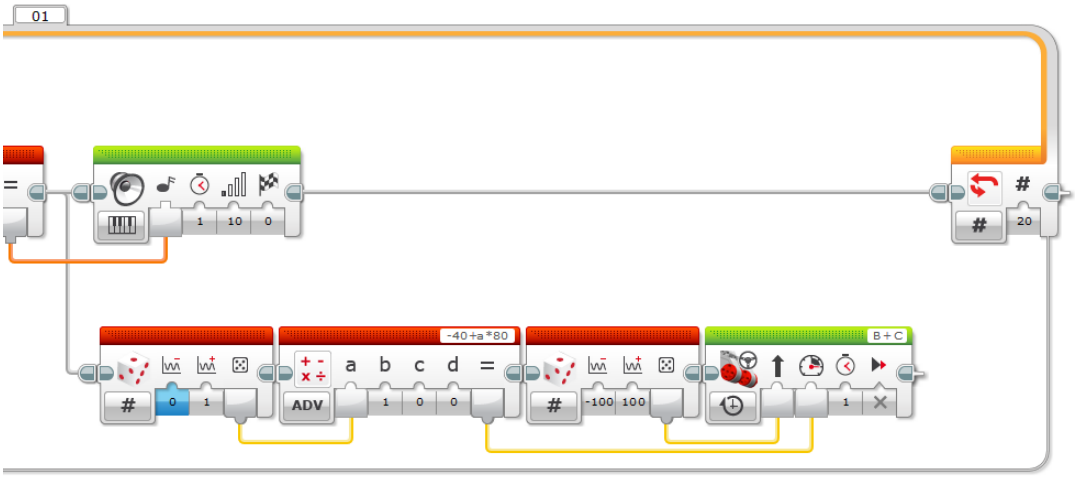


Figure 11. More random

4.4.2 Variant b: Simple waltz [easy/medium]

In this variant we would like to reproduce a waltz rhythm and something similar to the corresponding dance by the robot. For the sake of simplicity, we organize the dance in a loop and every cycle executes the sequence of three components, each one, like before, made of two concurrent actions, one sound and one motion of the same duration. For rhythmic reasons, along the main flow we insert a little silent pause after each motion, while we generate randomly the first note of each triplet of notes between two choices to recall the typical accompaniment of this rhythm (fig. 12). If you prefer, you can ask the students for the first note to alternate between the two chosen (G4 and C5). In this case you need a state logic variable to change to the negated value at any repetition (fig. 13).

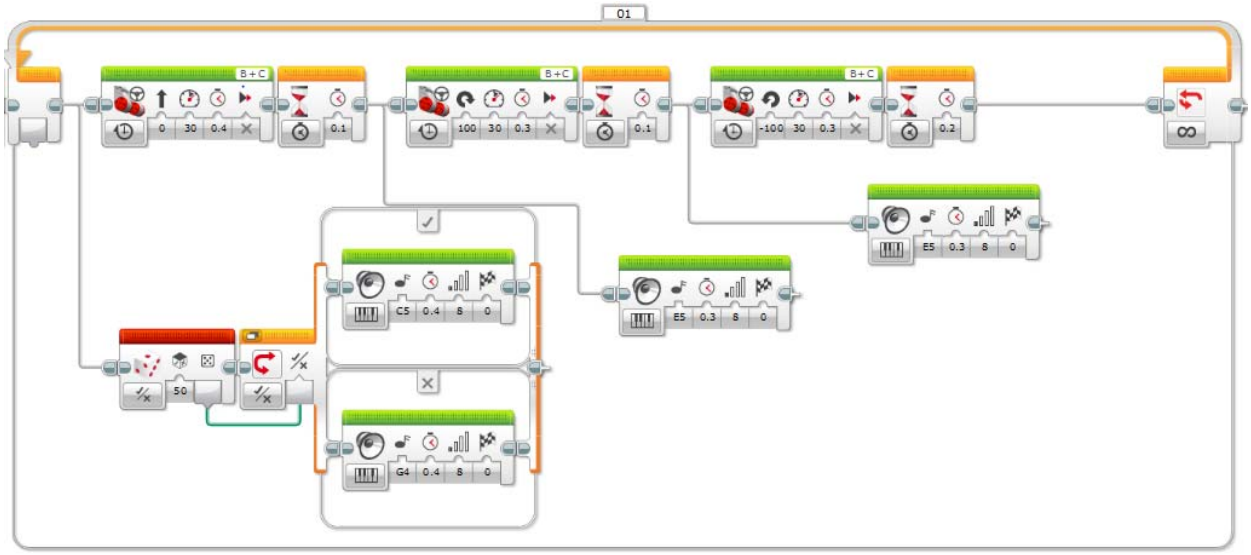


Figure 12. Simple waltz

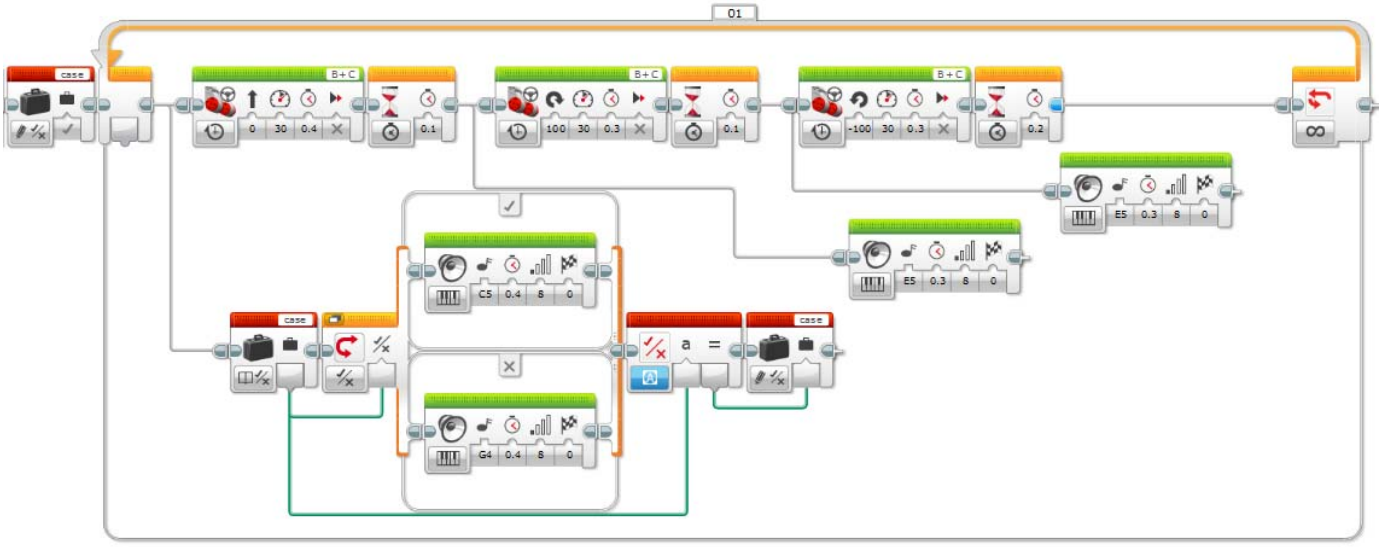


Figure 13. Simple waltz with alternative first notes

A third solution adds a small scale of notes after 3 triplets: these notes should be G4, A4, B4, an ascending scale with one full tone in between two notes. So knowing that 440 Hz is the standard frequency of A4, G4 has a frequency of $440/\sqrt[12]{2} \approx 391$ Hz and the distance of one tone corresponds to a factor $(\sqrt[12]{2})^2 \approx 1.12$. So you can insert the previous loop, executed only 3 times, within another loop and followed by a loop for executing the scale: starting from 391 Hz, the following frequencies in the sequence are obtained multiplying the previous frequency by the 1.12 magic number (fig. 14).

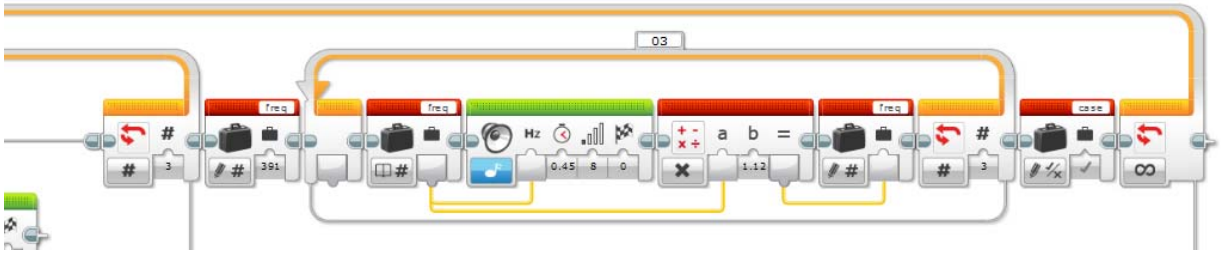


Figure 14. How to add a small scale

4.4.3 Variant c: Waltz [medium/difficult]

In this variant we would like to make a more explicit synchronization between two independent flows of execution, the one producing sounds and the one making the robot move (fig. 15). For this purpose, we use a synchronization logic variable (called *sync*), set at the end of every repetition in the first flow to enable the second flow to start its next repetition: for this purpose, the second loop, before repeating, waits for the signal represented by the change false->true, done at the end of the first loop, and then resets the variable. The waiting is implemented in a conditional loop where the *sync* variable is continuously read until it becomes true. Another logic variable (*case*) is also randomly set by the first flow to adapt the motion to the first note of the triplet chosen for each repetition: in one case it is a forward motion, in the other case a backward motion.

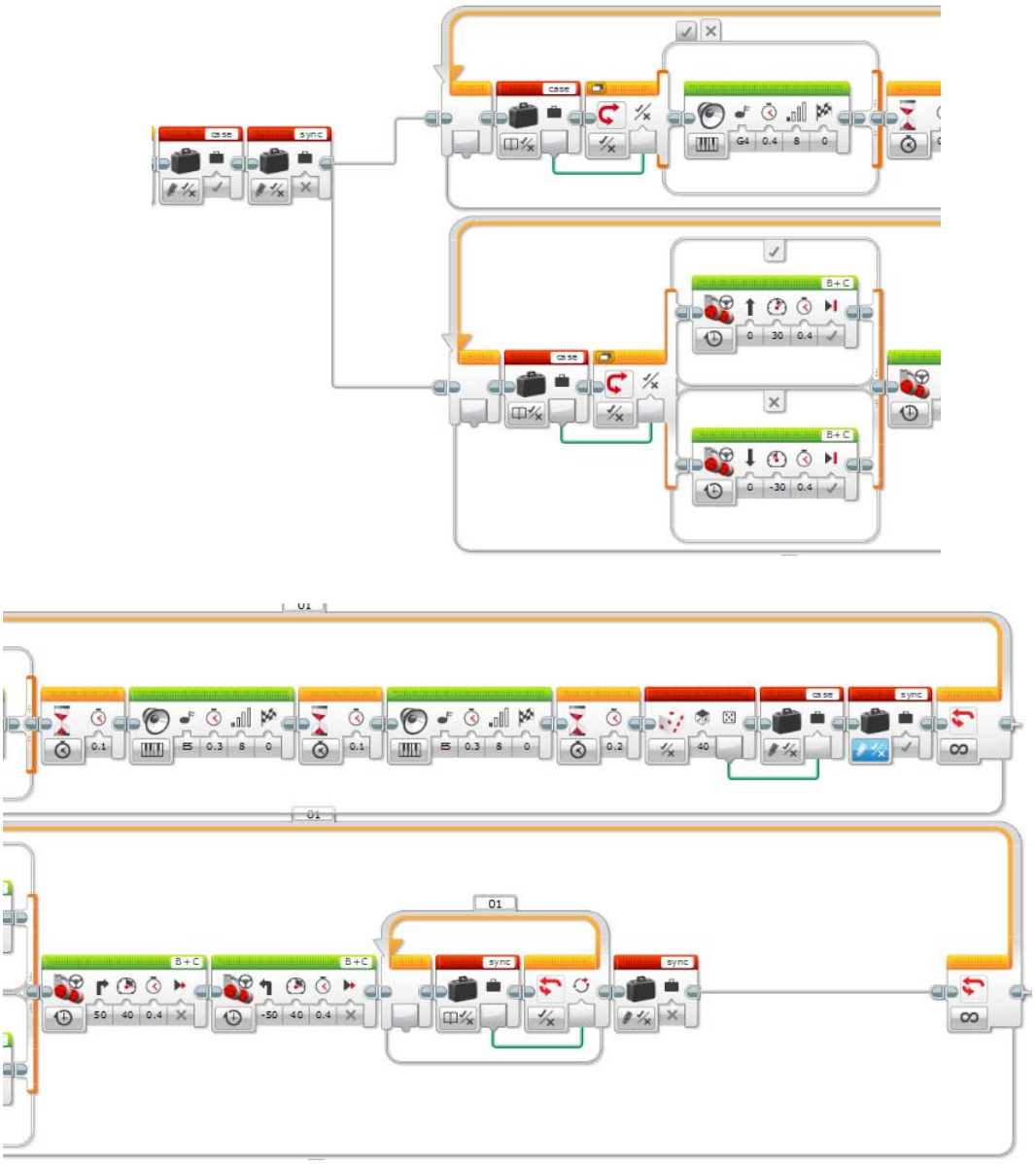


Figure 15. A more complicated waltz

4.4.4 Variant d: Play&Dance [medium/difficult]

If you desire to have a more flexible solution to reproduce a complete tune and dance you need to code in a file the tune, a sort of 'music sheet' that the robot can reproduce and on which it performs a synchronized dance. Though the EV3 firmware can reproduce a music file in a specific format that can be downloaded into the robot memory, there are two big limitations: the file cannot be longer more than some seconds of reproduction, and the robot motions cannot be easily synchronized with the music. So we suggest to provide a text file which contains the information useful to code the tune as a sequence of notes with no predefined length.

Now we describe how to prepare this input file. It is a simple text file, saved with a name with extension *.rtf*. In order to read (or write) such a file you must use the `File Access` block you find in the `Advanced` category (fig. 16). With this block you can read a string (*Text* option), long as the current line of text in the file, or a number (*Numeric* option) for which the reading includes a conversion from text to the internal representation of numbers. When you write something, a new text line is appended to the file; if initially the file does not exist, the first write creates it. After a sequence of writes the file must be closed with the *Close* option.

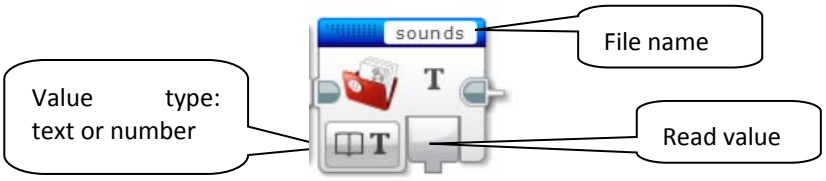


Figure 16. The File Access *block*

There is a technical detail to take into account: the text file format requires that each line is terminated as the Unix format (i.e. with the only character *newline*, that is LF in the ASCII table, as line terminator, and not with the usual Windows sequence of two characters CR-LF). If you don't know how to produce a first instance of this type of file with a normal editor, one possibility is to make the robot do it with the program of fig. 17. This program creates, and fills with three lines, a file named *notes.rtf* that you can upload to your PC and use as a skeleton to edit the 'music file'.

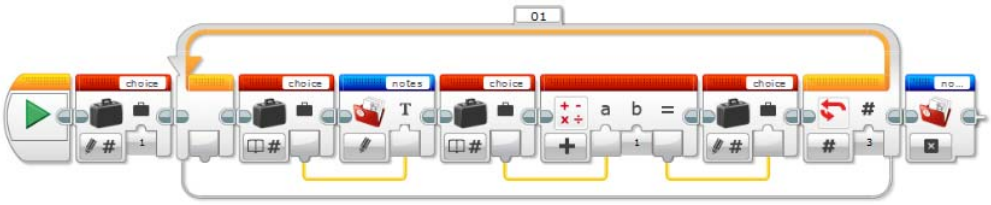


Figure 17. How to produce a text file in EV3

In the music file each note is represented by three ordered items, one for each line: a letter for the note, the optional sharp symbol (#) followed by a (mandatory) number 4, 5 or 6 for the octave, a (possibly) fractional number for the duration. The last triple must have a dummy note letter X and a duration value of 0. Separating the letter from the octave simplifies the process to associate in this experience the note letter with a specific 'dancing' action (seven notes, seven different actions, independently of the octave). Obviously, we have then to concatenate note letter and octave, both read as text, to prepare the parameter the Sound command is expecting. In fig. 18 this part of the program; *popcorn.rtf* is the name of the input file.

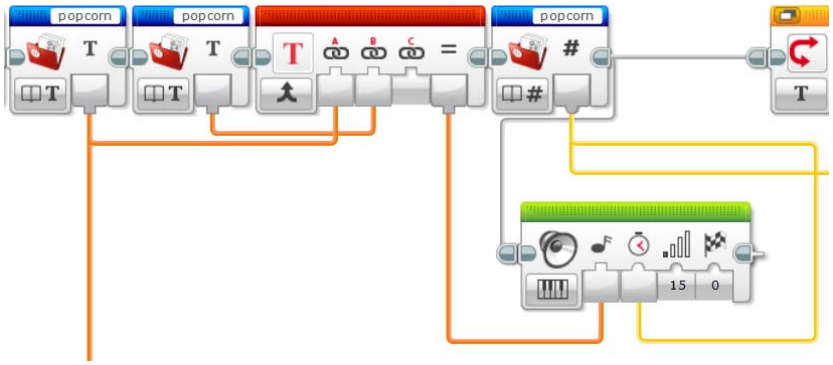


Figure 18. Preparation of the Sound *command's parameters*

Fig. 19 shows an example of the input file: numbers on the left are only line indexes, not inserted in the file body. To download the file into the project store in the robot, with the robot connected choose the robot memory browse function, select the project and click on the *Download* button.

```
 1  E
 2  5
 3  0.5
 4  D
 5  5
 6  0.5
 7  E
 8  5
 9  0.5
10  B
11  4
12  0.5
13  G
14  4
15  0.5
16  B
17  4
18  0.5
19  E
20  4
21  0.5
22  C
23  5
24  1
25  F
26  5
27  0.2
28  A
29  4
30  1
31  C
32  5
33  1.5
34  X
35  9
36  0
37
```

Figure 19. An example of input file

Now, the last part of the program associates actions with notes. This is not so difficult: fortunately, we have an option of the `Switch` command which is perfectly suitable for the purpose, the *Text* option (fig. 20).

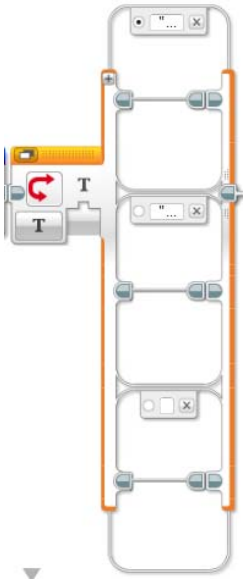


Figure 20. The Switch *command with the* Text *option selected*

When the *Text* option is selected, the block shows a number of choices which can be increased clicking on the small button '+' (*Add Case*) on the left top. Click it so many times to obtain 8 choices, 7 for each note (we ignore here the possible sharp modifier) plus 1 for exiting. Add the note letter within "" as the label of every choice and the associated motion command as its body. Add an "X" for the last choice and two block, a *Stop* command and *Variable* block to set the *exit* variable to true. This logic variable is used to signal the end of the loop. The whole solution is shown in fig. 21.

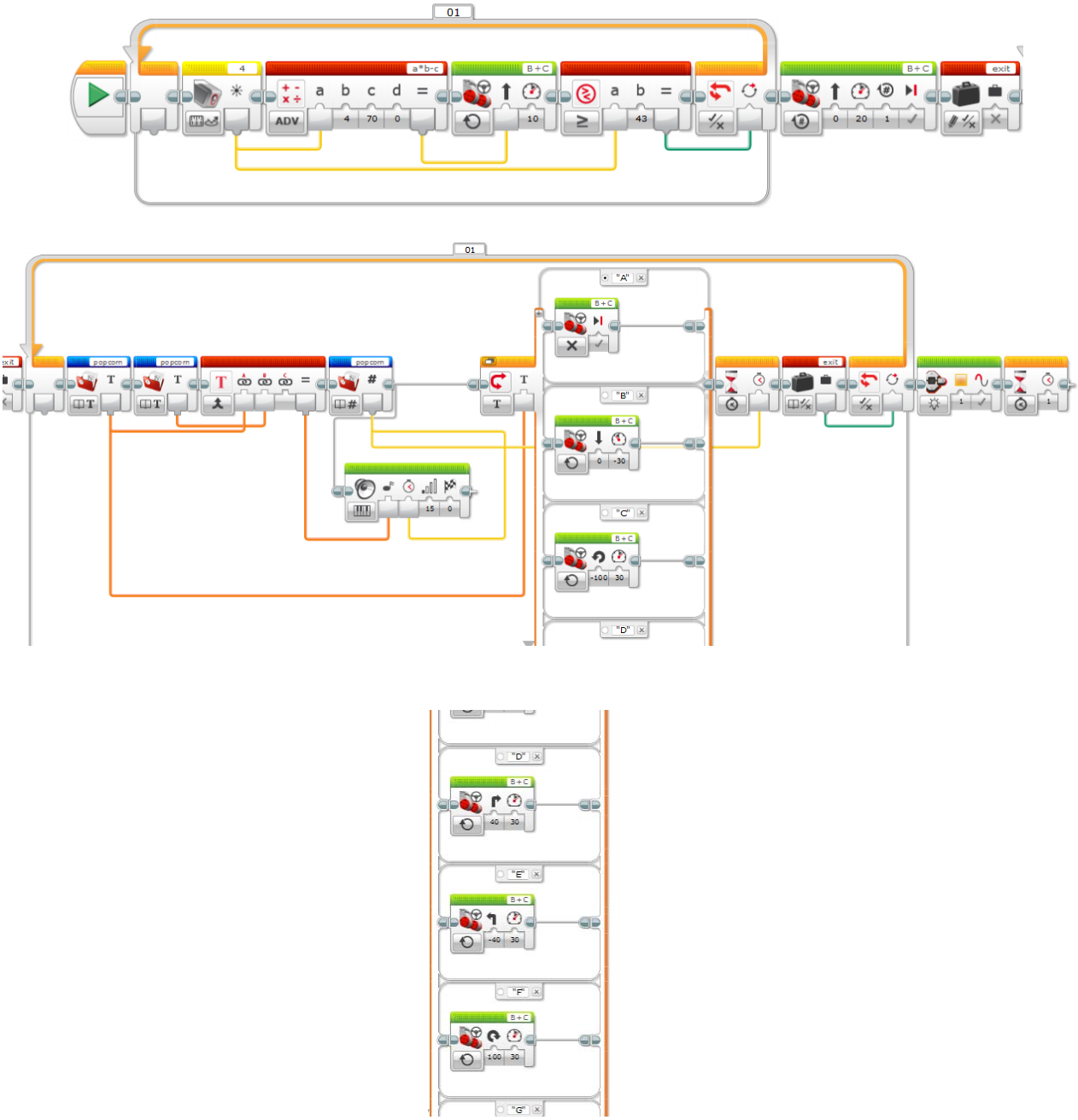


Figure 21. The whole play&dance program

At the beginning of the program we have added the previously described 'follow line' prologue. You can observe that the Sound command and the action section are again put in concurrent branches: in order to have a minimum synchronization between the two endings, a wait, long the same amount of time of the sound duration, is added after the action. The very last commands signal the program completion with a short flashing of the *Brick Status Lights*.

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**
Prepared a good scenario dressing the robot or adding suitable elements on the dance stage	
Properly used the "sound" command to reproduce music files or complex tunes	
Properly exploited the relationship between frequency and the musical scale	
Properly designed and realized a simple line follower using the color sensor	
Properly designed and realized an advanced line follower using the proportional control of steering	
Discovered and properly used the "random" command	
Discovered and properly used the parallel sequences feature to implement multitasked solutions	
Properly produced random sounds and synchronized motions	
Discovered and properly used the "File" command	
Properly coded a generic tune in a text file	
Instructed the robot to play and dance synchronously	
Properly used the text variant of the switch command	
Instructed the robot to play and dance taking the code from the input text file	