An introduction to Robotics with LEGO® MINDSTORMS (XII)

Social Use of LEGO® MINDSTORMS

By Koldo Olaskoaga

In July I had the opportunity to participate in the course “Caring robots”, organised by the universities of Windesheim Flevoland in Almere (The Netherlands) and La Salle - Universitat Ramon Llull in Barcelona. In this course, among other things, we discussed what activities with robots can be done to help in the rehabilitation of children with brain damage and in the improvement of social skills of children with Autism. La Salle - Universitat Ramon Llull is collaborating with Tufts University, Massachusetts (USA), Deusto and Comillas in the investigation that aims to determine if the use of robots in the rehabilitation of children with brain damage has advantages over more conventional methods.
One of the tasks that was proposed in the course, was the creation of an activity that would later be tested with brain damaged children in the Sant Joan de Déu hospital in Barcelona, principal collaborator in the project of La Salle.
In this article I will present the activity, concentrating the explanations on the programming task it involved.

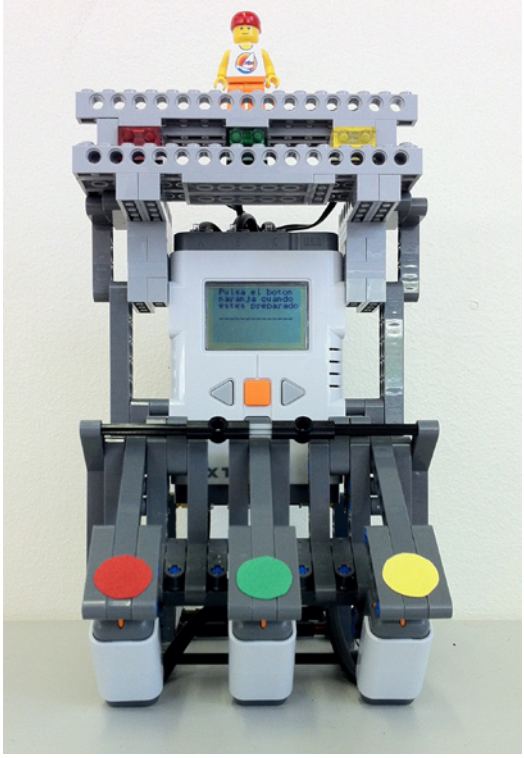
The activity

One of the ideas that was proposed was the creation of a game to stimulate the memory. After considering and analysing the possibilities LEGO® MINDSTORMS Education offers, the material that was available, it was decided that we would create a game that involved remembering a sequence of sounds. The robot should reproduce a sequence of up to four tones (one for each entry port on the NXT) and the child should repeat the sequence using the keyboard we were for create to this end.
It was considered that remembering a sequence of tones could be complicated for some children - due to the fact that they are sounds - so it was decided to combine them with some other kind of stimulus, in this case colours. In the LEGO MINDSTORMS Education set there are three lamps that can be fitted with three different coloured transparent pieces, which allows for the inclusion of three lamps (red, green and yellow) that can light up while reproducing the tones. Since there were three lamps and also three entry ports it was decided to reduce the keyboard to three keys.
What would happen if the child enter the right sequence? In addition to the applause and a smiley face on the NXT display it was thought that it would be nice to include some kind of reward to add to the fun. To this end a mobile robot with a light sensor was built, which would advance or move back over a white surface with black transversal lines; with a right sequence it would advance, with a wrong one it would go back. The objective of the game was to make the robot reach the last line.

The game console

When building the game console the requirements were the following:
- Three touch sensors with sufficiently big and separated keys to be used by kids.
- Three lamps with a coloured cap aligned with the keys.
- A structure that would allow for the console to be set on the table so as to allow easy access to the NXT screen and the NXT keys.

Since there were no additional LEGO parts in the three colours, stickers were used to identify each key.

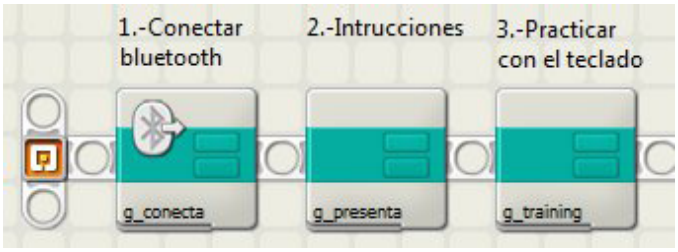


The program

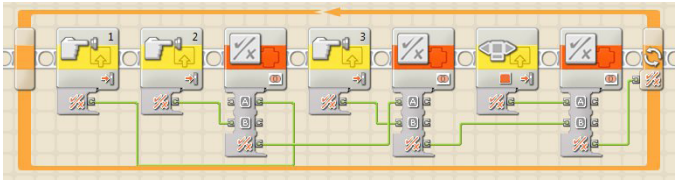
I will not go into every detail of the program, as this would make the article too long, but I will comment on some of the aspects I consider to be of interest.

The program of the console consisted of the following parts:
1. Establish a Bluetooth connection
2. Present the instructions
3. Practice with the keyboard: in order to get to know the keyboard. When pressing a key, a sound is reproduced and the corresponding lamp lights up. Upon pressing the orange button the program advances.
4. The game
    a. Choose a level: Training (a sequence of 3 tones/ colours), Basic (4), Medium (5) y Advanced (6)[1]
    b. The game itself: generate a random sequence; reproducing it; registering the key strokes; checking the result and giving feedback

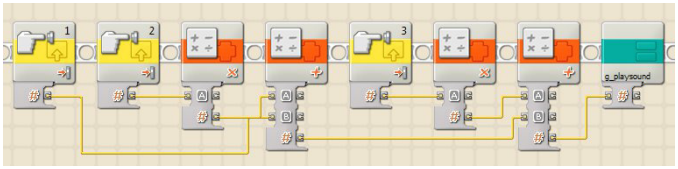
The program was built in a modular fashion, so each part could be tested separately before moving on to the next. New blocks were created whenever possible. The following image shows the first three steps of the program, each one converted into a new block.



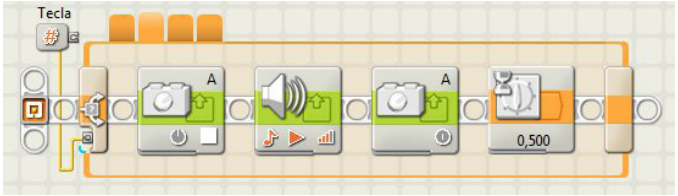
Let's leave the first two steps aside and concentrate on the third. In this step you can practice with the keyboard and check how it works for as long as you like. The program continues and starts the game when the orange button on the NXT is pressed.
To this end, the NXT needs to read three touch sensors and the orange button sequentially until one of them is pressed. It uses a boolean OR to determine if one of the sensors has been touched. If so, it exits the loop and reproduced the sound or continues with the program.



Once one of the keys or the orange button on the NXT has been touched, the corresponding note is played and one of the lamps lights up, or the program exits the practice stage.
This step can be done with a Conditional (If... Else...), but in order to avoid adding one inside another we converted the input into a number. 1, 2 or 3, depending on the key. This can be seen better in the following fragment of the program:

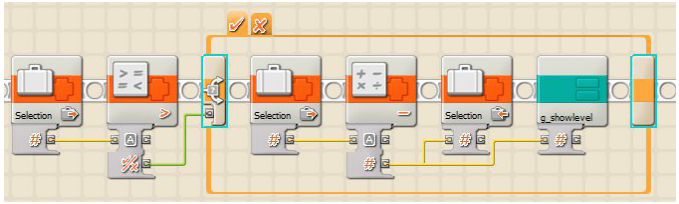


We take advantage of the fact that one of the outputs of the block is Read Touch Sensor, which gives a value of 1 if touched and 0 if not. Leaving the first value at 0 or 1, multiplying the second value by 2 and the third value by 3 and then adding these values we'll know which key has been pressed. In case if the orange button has been pressed the value will be 0. At the end you can see the block to make the sound and light the lamp which has been turned into a new block as it will be used several times in the program. The content is as follows:

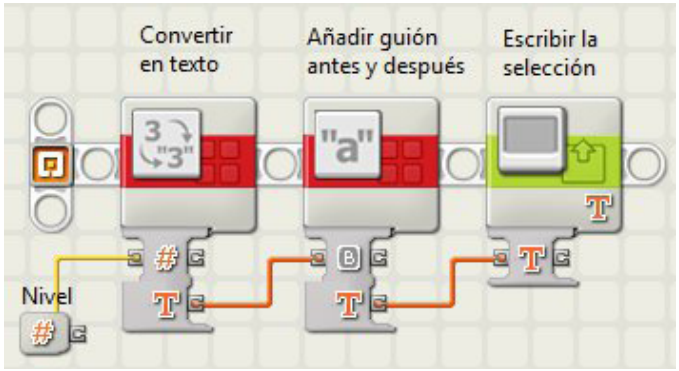


After practice it is time to play. First the desired play level must be selected. To this end the grey arrow keys are used to select the level and the orange button to confirm the selection and start the game. The steps to follow are these:

1. Show options
2. Create a variable to store the level and set it to 0.
3. Show the level that corresponds to this value
4. Start a loop that will not end until the orange button is pressed:
    a. Wait for one of the three buttons to be pressed (orange or grey arrows)
    b. If it is one of the arrow, add or subtract 1 from the variable and show the new selection (there are four levels and the value must be between 1 and 4). Care must be taken not to fall out of the range when adding or subtracting and that the result is never 0 or 5.



In the image you can see the code that runs when the let arrow is pressed. If the contents of the variable Selection is bigger than 1, one is subtracted and shown on the screen, otherwise it is left as is.
When representing the options and the current selection some precautions must be taken. The View block has a verification box which erases everything if checked. For showing the level a conditional could be used, but it can be done more directly using the blocks that allow text operations. When an option is changed, the numeric value is converted into text and a dash is added before and after (simply for aesthetic reasons). The resulting text chain is written on line 8, overwriting the previous selection.

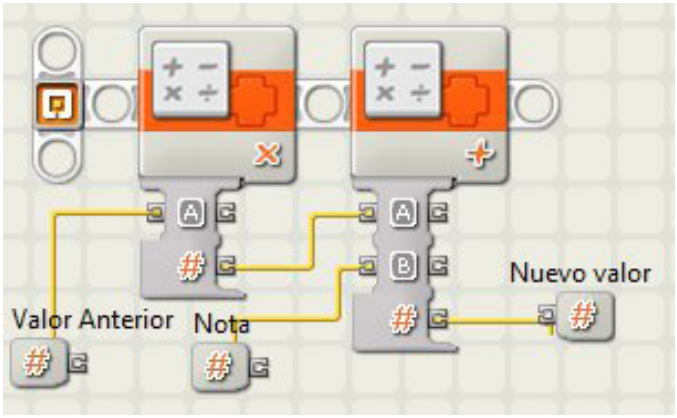


After selecting the level it is time to play, so the program picks a random tone, reproduces it (lighting up the corresponding lamp at the same time) and stores it. This is repeated as many times as the selected level requires.

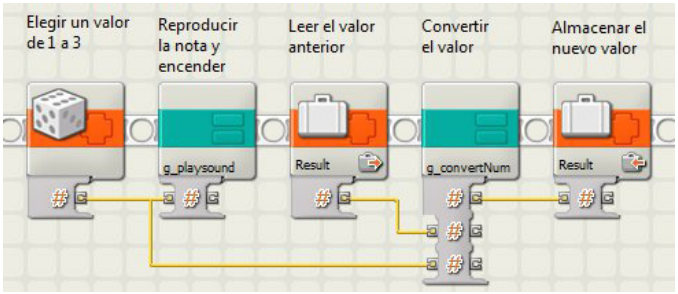
The most direct way of storing a sequence of notes would be in a matrix, but since NXT-G does not have standard support for matrices we chose a different method. We used a single numeric value in which the tone (a value from 1 to 3) is stored in the position of units, tens and hundreds. To do this, after generating the tone with the Random block (with a value of 1-3), the following calculation is carried out:

New value = Tone*10 + previous value

[1] g_GenerateSo: This block generates the sound sequence. The entry level is 1-4 and the exit level is a number that represents a sequence of between 3 and 6 sounds/colours.



So for each note the steps shown in the picture are carried out.

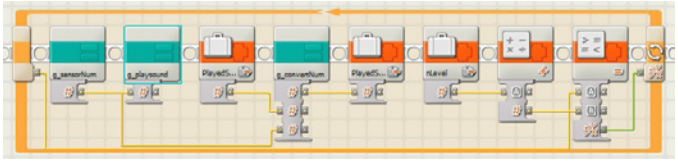


This is where the g_playsound block, which we saw earlier, is reused and a new one is created to convert the value as explained above.

When the user enters a sequence via the keyboard the same operation is carried out, so in the end the value that is generated can be compared to the stored value.

The loop that allows the user to introduce the sequence is as follows:

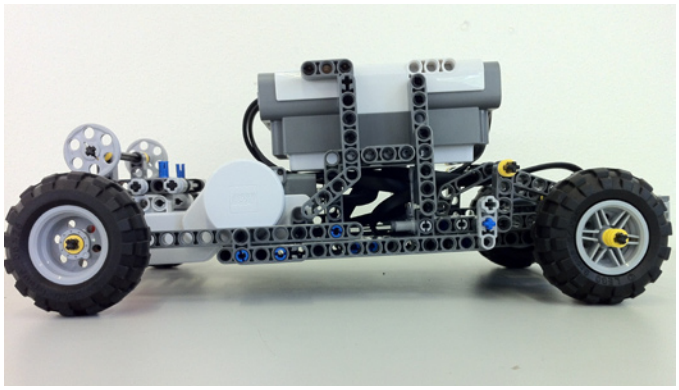
1. Wait for a key to be touched and convert the value into a number from 1 to 3 (g_sensorNum)
2. Reproduce the sound ad light up the corresponding lamp (g_playsound)
3. Register the choice as seen before (bloque g_convertNum), in this case the variable is PlayedSound.
4. Add a unit to the variable nLevel (which stores the level with a value of 1 to 4) and compare it to the number of completed loops. If the result is True it means the sequence has ended.



This loop needs to run more or less times depending on the level: the value that represents the level +2, that is to say, between 3 and 6 times. In order to keep count, instead of creating a variable that acts as a counter we used the counter of the loop. The connector of the counter is shown on the left of the loop (you need to check the box for the corresponding entry in the Loop configuration panel to see it). The value obtained from this connector is the number of completed loops. For this reason the value of the counter is compared to the value of nLevel +1.
Now what remains is to compare the reproduced sequence to the one generated previously. If they match, a smiley face is shown on the screen and you hear applause, otherwise, a sad face is shown and an error sound generated. Additionally a True or False signal is sent to the vehicle by Bluetooth.
The original program allowed for 5 opportunities to get the vehicle to the finish line.

The vehicle

The vehicle was very simple, as the only requirement was the possibility to advance or go back to the next line. We could even have used a single motor and a sensor looking down.



The program is waiting to receive a True or False message from the game console. After receiving it, if the value of the message is True it will advance till the next line, while if the signal is False is will go back (unless it is at the starting line). The program registers the number of tries and the position, so when 5 attempts are reached it goes back to the start position, whether it has reached the finish line or not (in the first case only after reproducing a triumphant sound).
It is important for the robot to control where it is regarding to the starting line, that is to say, whether it needs to cross it first or not. This was something that gave a lot of errors when programming the vehicle.

Putting it into practice

The activity was tested at the Sant Joan de Déu hospital in Barcelona with several children with Brain Damage. While the test conditions were insufficient to draw conclusions, it was observed that little children focussed completely on the vehicle, forgetting the other part of the game. On the other hand, in some cases three tones/colours were too much to start; two tones and colours would have sufficed.
From a human point of view it was a very enriching experience, thanks to the collaboration of the hospital and the participating families.

Possible improvements

- Use of the keys to control the menu with less able hands.
- While this was not considered originally, the mobile robot could pick something up and bring it to the user as a reward for finishing the exercise.

This activity was the result of teamwork in which the following people participated: Chang Long Zhu, Juan Pablo Forero, Louellen Palm and myself.
#