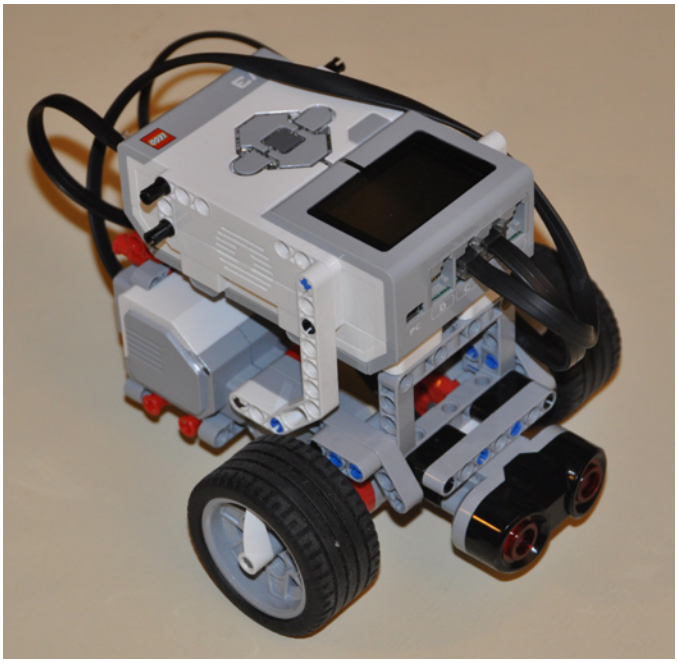
An introduction to Robotics with LEGO® MINDSTORMS (XVII)

Alternative Programming Languages for the EV3: RobotC

By Koldo Olaskoaga

Thanks to the contributions of the user community and commercial initiatives, the LEGO® MINDSTORMS NXT and EV3 can be programmed using many different programming languages, both graphical and text based. While the LMS EV3 ecosystem isn't on par with that of previous versions yet, there are several options available aside from the official EV3-G environment. This article will be about RobotC and future editions of HispaBrick Magazine® will talk about others.

I will use the basic robot of the Education version of the EV3 for the examples, but other robots with a differential drive will offer a similar behaviour.



RobotC

RobotC is a programming environment that is based on C and designed to be used in education and educational competitions. It offers a friendly environment for those who use C for the first time.

After a number of beta versions that were made available for testing, the first official version of the new RobotC 4.x for LEGO MINDSTORMS EV3 was made available at the end of August, after having been made available for other platforms as well.

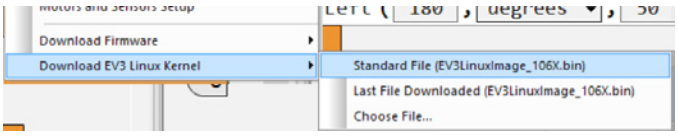
This new version not only supports the NXT as well as the EV3, but includes an important novelty: a graphical programming language that facilitates the transition from other graphical languages to text based C.

The RobotC Firmware

In order to be able to execute the programs that are downloaded to it, the EV3 needs to have a small firmware program installed. This way the EV3 can understand and execute the programs it receives. Updates to this firmware are published periodically.

Since RobotC is not compatible with the original LEGO firmware, the RobotC firmware needs to be on the Ev3. However, the incompatibility doesn't exist in the opposite direction, meaning that the RobotC firmware can execute programs created in both RobotC and EV3-G.

For this reason, the first thing you will need to do is install the RobotC firmware. To do so you need to go to the EV3 Linux Kernel from the Robot menu, and start the firmware download. After this the EV3 will be ready to start working with RobotC.



The Graphical Programming Environment

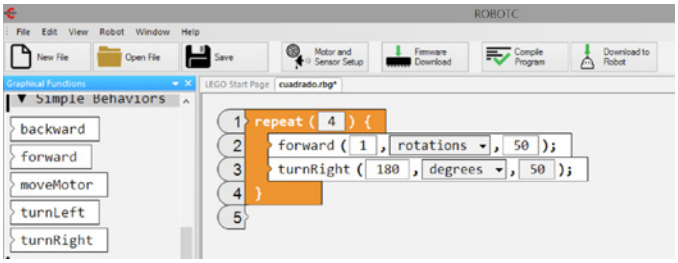
This new programming environment is reminiscent of Scratch, basically due to the control structures, which are orange and open up to contain the instructions they need to execute. The instructions are places easily using drag and drop.

This graphical mode uses a functions pallet based on what is called Natural Language. This is a high level language, meaning a language similar to what people use to communicate. Between the groups of functions the simple behaviours stand out: functions for going forward and backwards, turning left or right and turning a motor. These functions make it easier to learn and require a specific configuration of the motors and sensors to work as expected: left motor on port B, right on port C and the ultrasound sensor on port 4.

My First RobotC Program

Lest try to program a simple task: letting the robot travel in a square and stop at the starting point. The necessary steps can be summed up as follows: repeat 4 times forwards and turn right.

Let's see how to create this program in RobotC. Remember we need to use the default motor configuration, namely left motor on B, right motor on C.



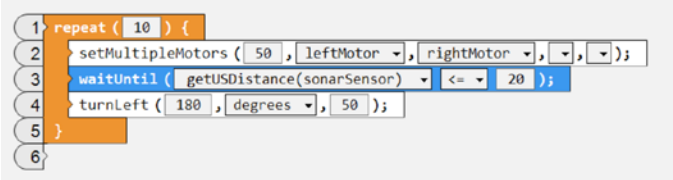
In the image you can see the program, with the function pallet to the left. In this example we use the repeat structure as well as two programming blocks: forward and turnRight. The third parameters in each of these functions is the power level, in this case 50. Because the functions in this group use similar behaviours, programming becomes relatively simple.

If the robot were to have been built with WeDo, we would be able to control it with Scratch using the following program or something similar. You can observe the similarities in the graphical design, although in this case the program is simpler in RobotC.

This is because in WeDo we use Technic motors which, when used side by side, need to be programmed to turn in opposite directions to move forwards.

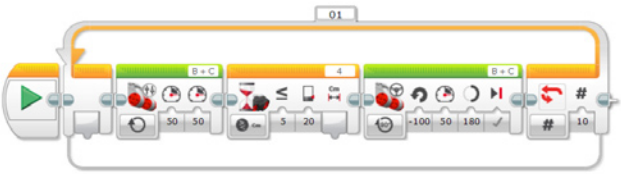
Robot with Ultrasound Sensor that Avoids Obstacles

In this new program we are going to combine the use of motors and a sensor. The objective is to make the robot go forwards in a straight line until the distance to the obstacle is equal to or less than 20cm. Then it needs to turn and repeat the previous step so that after detecting an obstacle 10 times it will stop. The sequence of steps is quite simple; repeat 10 the following: times start the robot to move in a straight line, wait for an obstacle and turn.



The program is simple and easy to understand In this case the forward block that was used in the previous case can't be used since that block only allows you to program a specific distance or time. What we need here is a block that starts the motors and allows the program to continue with the next step. To this end we will use the setMultipleMotors block that allows the motors to be started simultaneously.

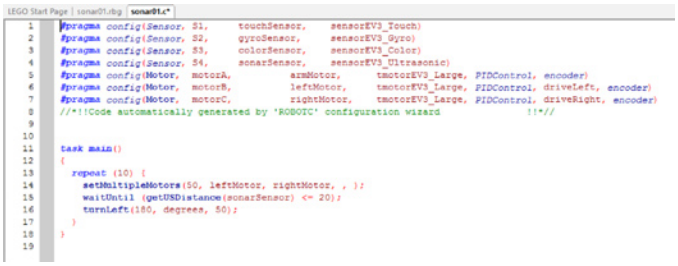
After starting the robot the next instruction, in blue, tells it to wait until the US sensor indicates a distance less than or equal to 20cm. Next it turns and we start again.
The corresponding program in EV3-G would be as follows:



As you can see, taking into account the difference in graphical design, you can establish parallelisms between the programs.

Converting a Graphical Program into Text

One of the interesting characteristics of the new graphical interface of RobotC is the help it provides in transitioning from a graphical to a text based language. It includes a tool that allows you to convert the graphical program into a RobotC text program. For example, using the tool Convert Graphical File to text in the View menu, the previous program turns into this:

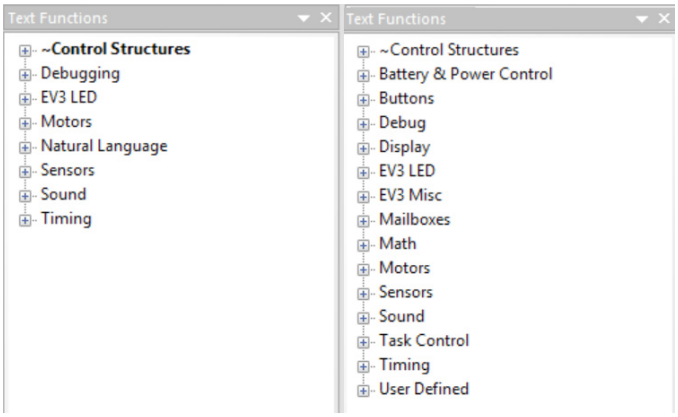


The top part of the code is generated automatically by RobotC depending on the motor and sensor configuration. The program itself corresponds to the code between the lines 11-18. In this case, due to the combination of a simple program and the use of Natural Language, both programs are quite similar, but in text based programming syntax is essential and forgetting to close a line with a ";" generates errors.

The Text Based Environment

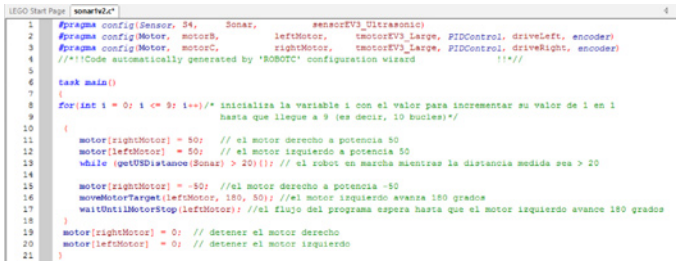
Once you are familiar with the graphical environment it is time to transition to text based programming. Here also there are two different programming options, natural language, which is also used for graphical programming, and text based RobotC. In the second mode there are three levels, basic, expert and super user. Basic mode limits the available functions as well as the available options and preferences. Expert mode gives you access to the full functions palette, while super user mode opens up all possible RobotC functions.

The following image shows the functions that each mode offers: natural language at the left and text based expert mode at the right.

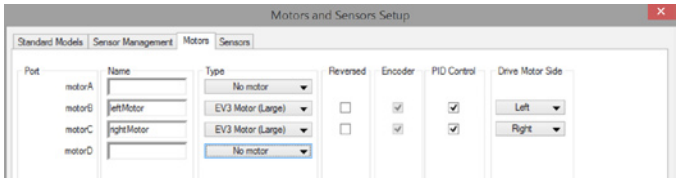


You can see the differences between the two function pallets; if you use natural language it is going to be easier to create a program, but access to all the functions in RobotC allows you to create more sophisticated programs.

The following is the previous example in RobotC text mode. You can see the motors are controlled individually and that instead of the structure repeat for a number of times, the For structure is used, which is the standard structure in many programming languages.

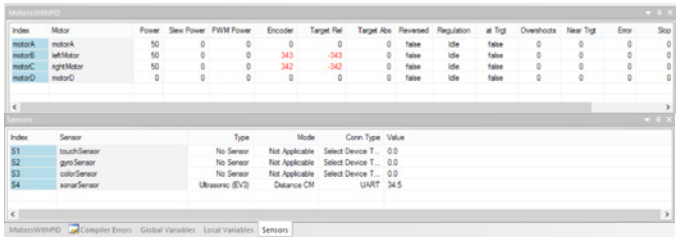


When programming in text mode, motors and sensors need to be configured before you start. The following image shows a configuration window for the motors with different options.



The Debugger

If things don't work the way you expected, RobotC has a debugger that allows you to observe how the motor and sensor values, variables, timers etc. change in real-time, so you can easily identify problems. Add to this the possibility of running the program step after step, either with the robot connected by USB cable or wirelessly.
In the following image you can see the state of the motors and sensors during the execution of the previous program.



To summarise, RobotC is a commercial programming environment with a graphical interface that provides an easy introduction and a text based programming language that allows you to use the full potential of the EV3. It is used in several robotics competitions, like FTC and VEX Robotics. There is an official forum that provides user support, mainly in English. Updates are published periodically.
#