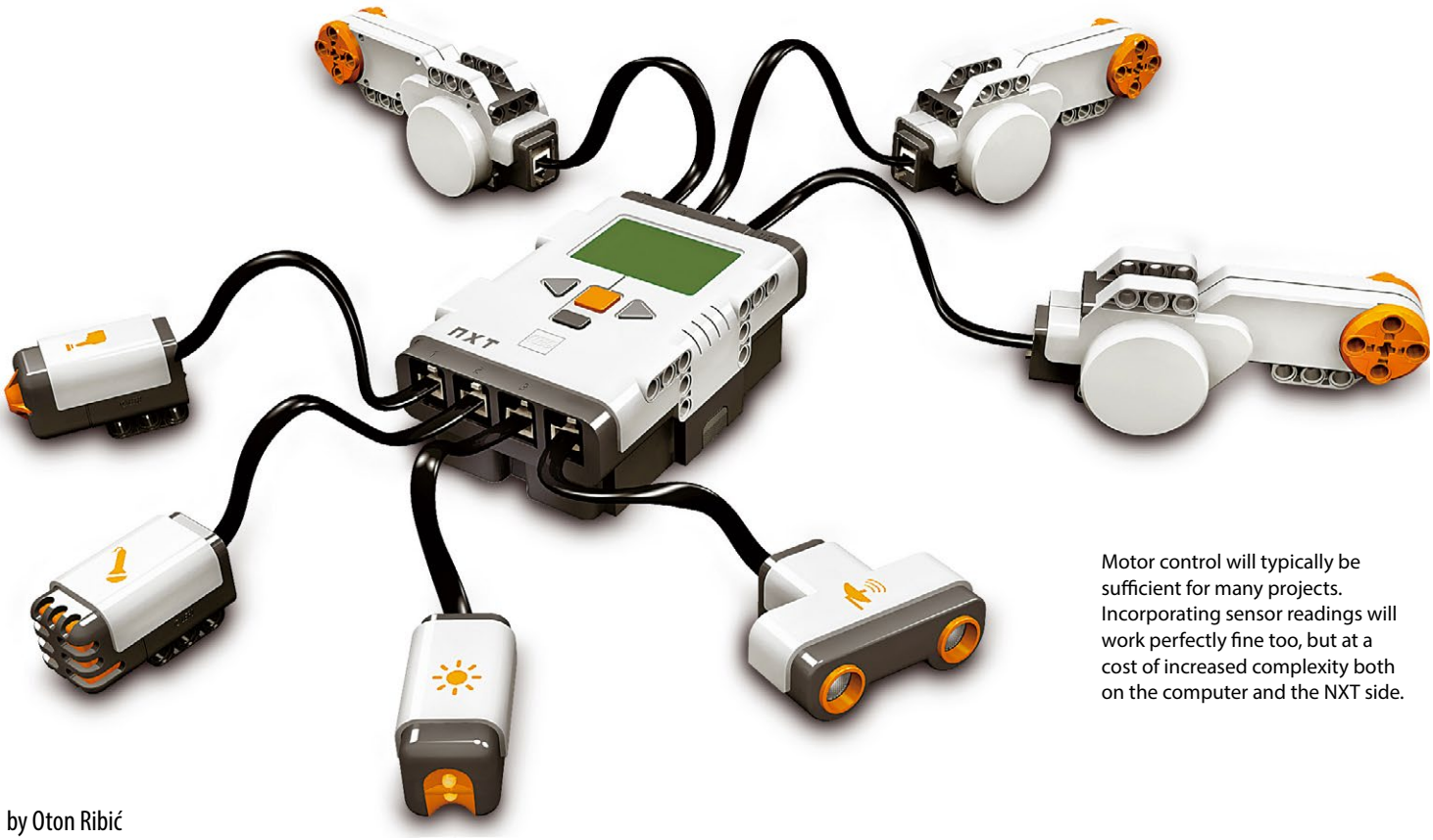
Back to the good ol' NXT



Motor control will typically be sufficient for many projects. Incorporating sensor readings will work perfectly fine too, but at a cost of increased complexity both on the computer and the NXT side.

by Oton Ribić

Having navigated through the tricky waters of the EV3 control via Bluetooth in previous articles, in this final episode of the already lengthy Mindstorms Control series we will get back to the somewhat aged, yet still very much vital, NXT. It supports Bluetooth communication as well, and if your hardware allows for it, there is no difficulty in combining NXT and EV3 Smart Bricks at once, or even multiples of each type simultaneously.

First of all, we should emphasize that the NXT is a somewhat different ball game than the EV3 when it comes to wireless control. Whereas the EV3 allowed us to compile messages to make it directly "obey orders", in the case of NXT it is simpler to communicate with a dedicated program running on it, which listens to the incoming data and executes the commands accordingly. This makes things more complex, as the program has to be running on the brick, but it makes the communication itself a little easier. Of course, one may decide to forgo the entire idea and flash the NXT with firmware to allow direct control–but that is a rather complex approach. As said before, in these series we will be focusing only on the stock firmware.

Program listening and executing on the device-side

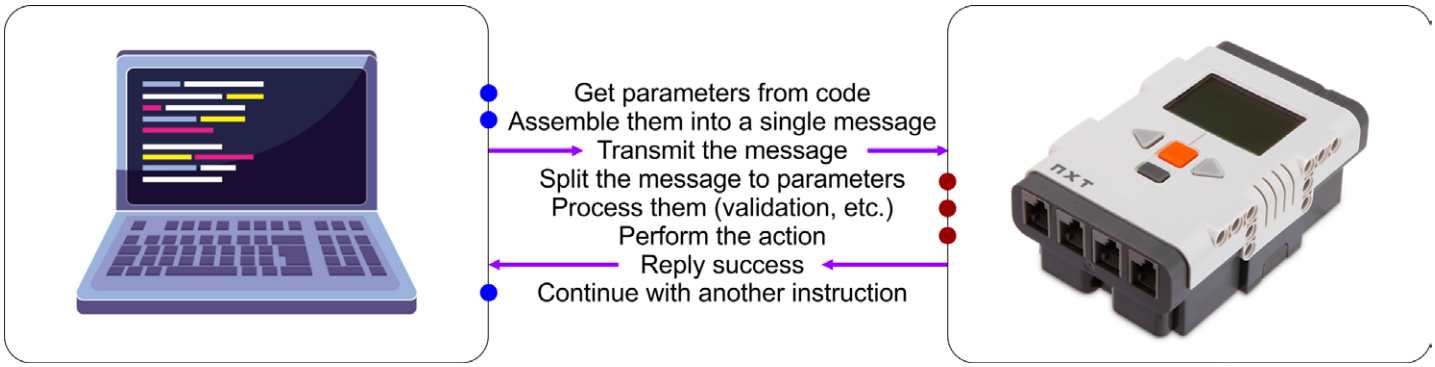
Therefore, the first step will be to set up a program aboard the device. You may opt for a general-purpose one, or for one that specifically does what you require for the build in question. Here we will go for the general-purpose version, and you can easily adapt it further for different needs based on these principles.

When controlling motors, there are three essential values we need to convey to the NXT Smart Brick for each movement: the motor to be used, the amount to turn, and the power to apply while turning. In theory, one could argue that direction is the fourth parameter, but we can embed this into the number specifying the amount to turn, by using either a negative or a positive number.

Assuming we load these values in the mentioned order, a program outline would look something like the following:

- Listen to the Bluetooth until messages are available
- Load a "triplet" of values from Bluetooth–Motor, Amount and Power
- Assign the Motor value to the corresponding parameter of the Motor instruction in the NXT-G
- Check if the Amount is below zero, and assign the result to the NXT-G Motor instruction's direction parameter
- Take the absolute value of Amount and assign it to the angle parameter of the instruction in NXT-G
- Assign the Power value which is a percentage to the corresponding parameter of the NXT-G instruction
- Perform the rotation itself and wait until finished (or not–if you want to use a special parameter for that!)
- If you want to be able to wait for the instruction to finish, send a message back via Bluetooth
- Repeat from the start unless an instruction for termination is received, which is optional

To give it a more understandable shape, this is an NXT program that listens to the Bluetooth, and when it receives values, e.g. 2,-720,50, it rotates the second (B) motor two turns in the reverse direction at half the power. Then it continues listening for a new triplet of numbers, specifying the new instruction.



A simplified sequence between the computer and the NXT, communicating via Bluetooth

This is all that's necessary for the remote control to work from the device side; just connect your computer to the NXT Smart Brick via Bluetooth as described in the first articles in this series, and start the newly-made program.

Control from the computer

Now we need to switch over to the computer side and set up a program that will transmit the three values required to perform a motor instruction to the NXT, and optionally wait for the acknowledgement.

This time, the assembly of the messages is much more straightforward than was the case with the EV3. Essentially, we will need to encode each of the input number parameters to a 4-byte code, representing a floating-point number. Check the same technique discussed previously in the case of the EV3, but as a quick reminder, for Python language for example, you can use "struct.pack('f', number)", which returns the necessary 4 bytes. Then, attach a zero-byte at the end, and the following byte sequence to the start: 5, 0, 0, 19, 10, 0, 1. This is a header telling the NXT Brick what is actually being communicated.

You should end up with a 12-byte sequence for each encoded number. E.g. encoding the number 720 should yield 5, 0, 0, 19, 10, 0, 1, 0, 0, 52, 68, 0. Finally, just assemble the final message by putting the three 12-byte codes one after the other (concatenate them) in the required order. The 36-byte message specifying the motor, the turning amount and the power can then be sent off to the NXT.

How to tackle synchronization

The procedure described above should make your NXT turn as instructed. However, the next step on the computer side is to get notified once the NXT has actually completed its task, so the next one can proceed. Previously in the NXT program you may have created a reply message for the computer–and now the computer is the one that has to react to it.

Regardless of what computer language you have decided to use, keep firing away the byte sequence 5, 0, 0, 19, 10, 0, 1 towards the NXT, which checks the device's "inbox" for any awaiting messages and transmits them if found. Repeat the process, and fetch any messages when they arrive. Typically just a simple reply by the device could be enough, but you can go a step further and have the NXT send a specific value, e.g. a string "DONE" or "Acknowledged", to be doubly sure. Then, on the computer side, you should disassemble the message received from the NXT, and check whether the specified string is contained therein. If yes, all is well; if not, you know that something on the NXT side went awry.

Nothing speaks against connecting to, and controlling multiple NXT and EV3 bricks simultaneously. But keep in mind you will have to configure serial ports for each such Smart Brick individually

Expanding further

This example as such should give you a general idea of how the communication works, how the NXT interprets the message, acts upon it, and reports back when finished. For basic projects where you require controlled motion, this should be more than enough. But if you need more, let's dive into several further points that may be useful or uncover some shortcuts for you.

It may be sensible to take care of at least minimal data validation on the NXT side. If the given motor power parameter is too low to even turn a motor, or if it is blocked from turning to the required extent, the instruction will essentially never finish, and your entire fine-tuned system will get stuck. Therefore you may want to add a threshold, or expand your NXT program to report back a special message (read by the computer) if a given timeout period has been exceeded, and the instruction aborted.

If your motors' motions seem to be correct but you think wrong ones are being turned, keep in mind that the motor A is really identified under number 1, whereas in many languages you may specify it as zero, which will not work.

This communication scheme reveals how the further expansion of functionality should work. You can expand both sides to communicate four parameters, among which the first one would be the type of instruction (motor, read sensor, or anything else), and the remaining three would serve as parameters for that very type of instruction. This requires a much more complex program on the NXT side, but it is doable.

Or you can try a middle line, and for your specific project set a rule, e.g. if the specified motor number is 10, instead of turning the non-existent tenth motor, the NXT will actually interpret it as reading the distance sensor and sending it back in the reply value–which of course should be equally prepared and parsed on the computer side.

In any case, regardless of whether you opted for the EV3, NXT or both after all, we will not even try to fool you into thinking you will not need to experiment a bit. But it will be fun, and we can testify from our own experiences that it is not really that difficult once you get the grasp of the basic concepts. At that point, the feeling of being able to control multiple motors and sensors via a program running on your computer will reveal an entire new horizon of possibilities. We're looking forward to checking out your advanced robotics stuff!