Take Control of your MINDSTORMS™ bricks!

By Oton Ribić

While there is no doubt that the original LEGO© method of programming MINDSTORMS bricks via 'blocks' is pretty clever and fun, many builders who dive into more complex projects soon find out that this can become impractical, with structures spanning over hundreds or even thousands of modules. In such situations, controlling programmable bricks (pBricks) via a standard programming language is actually easier.
Many solutions have been put forward to achieve this, including custom pBrick firmware and entire programming environments, resulting in interesting and impressive projects. They serve well, receive regular updates, and are a good start. However, for those who like to control things at a more technical level, or at least have some insight into what a communication and 'ordering' system for pBricks looks like, we will reveal some of these in this mini-series of articles. We also assume the user doesn't want to move away from the stock NXT and EV3 firmware.
Keep in mind that for the huge majority of purposes, sticking with the original MINDSTORMS programming environment, or some custom version of it, is perfectly fine; going deeper than this is usually a matter of curiosity, not necessity. Therefore, the following may serve as documentary trivia as well as for practical purposes. If you are interested in the latter, it's worth mentioning right now that this can all be fairly easily implemented in any general purpose language (e.g. it was done by the author in Python, and by another LEGO® fan in C++).

Communication dissection

Let us begin by digging into the methods the pBricks use to communicate with the external world. Although USB can be used, it is actually more common and usually more practical to use Bluetooth, and there are also a few more exotic options. But in all cases, we can begin by stating that all communication happens as a set of digital packages (literally collections of bytes) being sent to and received from these devices, via any of these communication methods. The aim of anything used to control MINDSTORMS systems is to be able to prepare, send, await, receive and interpret the messages containing these packages.

These messages contain all the information that is exchanged by the pBrick and the controlling device – in the most common case, a computer. Sometimes they send some numerical data, in other cases instructions, occasionally asking the recipient to send some data in return (e.g. checking the value of a sensor). And there is also a system of responding to such messages with confirmations that they have been received properly, which is essential if we intend the system to work reliably.

Bluetooth is a practical and simple way to connect computers or other devices to programmable bricks. Older devices usually need a dongle to enable it, but fortunately, it is almost always built-in with newer devices.

NXT vs EV3

The overall concept of exchanging such messages with packages of data applies to both the NXT and the EV3. However, there is one segment where they differ, which will become even more important later when we get into the message construction. Namely, EV3 with its stock firmware allows the controlling device to send instructions directly, regardless of the state the pBrick is in, as long as it is at least turned on, and execute them. Any 'block' we can program in the official software can just be packed up in a message and sent for the EV3 to execute.

On the other hand, the NXT with its stock firmware does not allow that so easily, at least not without using some special tricks which we will not be getting into in these articles as they would just grow into a huge book. But what the NXT allows just fine is running a program aboard itself, programmed in completely regular NXT-G, which receives Bluetooth messages and interprets them in the required way. For example, correctly interpreting a message stating it is an instruction for rotating a motor, then specifying which motor should rotate, then the amount to turn, then the direction, and finally the power. Of course, these messages need to be carefully constructed to be interpreted properly, but that is not really as difficult as it sounds, at least not as long as only a few of the most common commands are used.

But suffice it to say for now that these two MINDSTORMS packages differ in that respect. EV3 is more flexible and allows more direct control, but is also more complex. NXT is a simpler case, but not as powerful. Still, for the most common case of rotating motors by controlled angles, it works just fine.

So much for the introduction – stay tuned for the next article, when we will get down to constructing simple messages and then sending them over to the pBricks.

Slightly older, the NXT is simpler than the EV3 but still allows most functions to be easily controlled with standard firmware.

The EV3 is newer and allows deeper control of the programmable brick, but is more complex too.