Lesson

DATA INPUT VIA PROXIMITY SENSORS

CREATED BY

Wolfgang Ettenreich
wolfgang@holismatrix.org

TOPICS

Computer Science, Programming, Mathematics

AGES

Grades 6-8

METHOD

OzoBlockly

DURATION

2-6 Hours

by Wolfgang Ettenreich

Data Input Via Proximity Sensors

Hack Creativity - The Ozobot Evo has four proximity sensors which are primarily used to help the robot to move around obstacles and to avoid any collisions. Are there other ways we could use these sensors? Yes there are and in this activity we will see how we can use them to enter data which we then can pass to our own programs.

There is a wide range on how you, the teacher/instructor can use this activity depending on the time available and the skills of the students.

Prerequisites / Preliminary Work

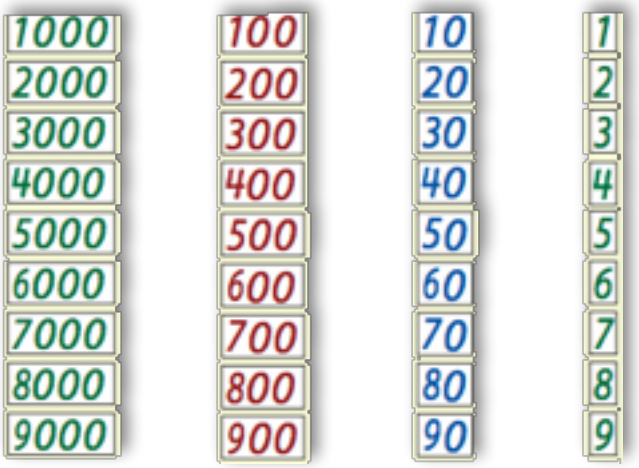
Listed below are five areas which each student needs to be somewhat familiar with in order to use the new input method.

- Understanding of the decimal system: 10-based, place value etc.
- Understanding of the binary system: 2-based (vs 10-based), place value etc.
- Ability of converting decimal numbers into binary numbers
- Intermediate level skills of coding with Ozoblockly
- Some familiarity with Evo's proximity sensors

The following sections provide some ideas/examples of lessons and activities for the students. Feel free to adapt any of these (or skip them) as it suits your class. The main idea is to demonstrate the students a "***creative hack***" and the value of it: being able to upload a program to the Evo just once and then run it with a variable input parameter as often as desired. To do this the students should be able to write their own (small) programs and call the input method as a function.

Binary Numbers

Let's take a brief look at the **decimal** system first. It is based on 10 and uses ten digits: 0, 1, 2, ... 9. Their place in a number determines the actual value - from right to left: units, tens, hundreds, thousands, etc.



Example:
Let's take a look at the decimal number 5318 and a variety of ways to write it.

- Five thousand three hundred eighteen
- Five-thousands three-hundreds one-ten eight-units
- 5 x 1000 (5 thousands) **plus** 3 x 100 (3 hundreds) **plus** 1 x 10 (1 ten) **plus** 8 (units)
- 5 x 10 x 10 x 10 **plus** 3 x 10 x 10 **plus** 1 x 10 **plus** 8.
- 5 x 10^3 **plus** 3 x 10^2 **plus** 1 x 10^1 **plus** 8 x 10^0 (per definition 10^0 equals 1).

The **binary system** is similar but based on 2 and it uses only two digits: 0 and 1. Their place in a binary number determines the actual value: ..., 64, 32, 16, 8, 4, 2, 1.

Example:
Let's take a look at the binary number 10011 and ways to write it

- **1** x 2 x 2 x 2 x 2 **plus 0** x 2 x 2 x 2 **plus 0** x 2 x 2 **plus 1** x 2 **plus 1**.
- **1** x 2^4 **plus 0** x 2^3 **plus 0** x 2^2 **plus 1** x 2^1 **plus 1** x 2^0 (per definition 2^0 equals 1).
- The decimal value of this binary number is: 16 + 0 + 0 + 2 + 1 = 19.

Activity:
Chose a binary number with a maximum of 7 digits, e.g. 1011011. Let the students write the number in the 2 ways as shown in the example above. Are they able to determine the decimal value of the number?

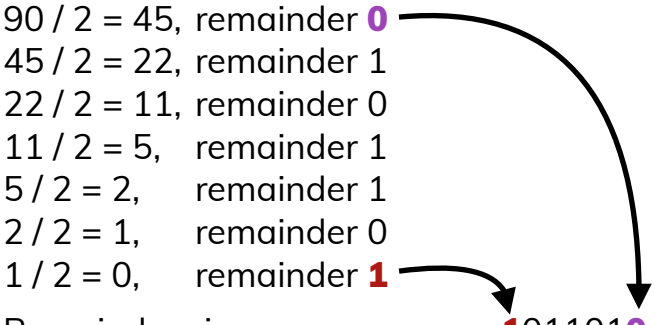
A really nice introduction to the binary number system can be found here:
https://www.mathsisfun.com/binary-number-system.html

Converting Numbers

Pretty much all numerical values we use for code blocks are decimal numbers, e.g. the movement block with the distance in mm and the speed in mm/s. Even though there are online tools to quickly do this job for us, it might be worth learning how to do this manually.

The technique is a repeated division by 2 and writing the remainders, 0 or 1, in sequence.

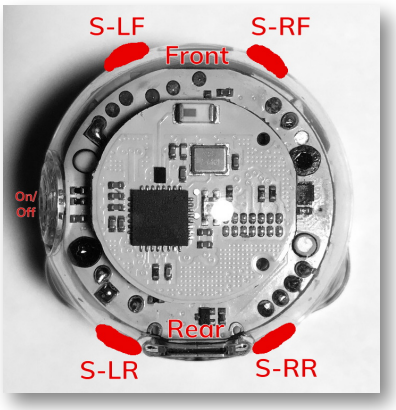
Example:
We would like to use 90 as the distance in mm for the Evo to move. To use our new input method we need to convert the number into a binary one first.



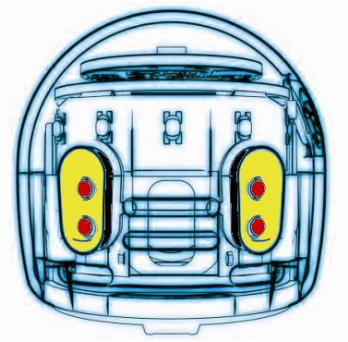
1 x 2^6 **plus 0** x 2^5 **plus 1** x 2^4 **plus 1** x 2^3 **plus 0** x 2^2 **plus 1** x 2^1 **plus 0** x 2^0 =
64 + 0 + 16 + 8 + 0 + 2 + 0 = 90 ✓

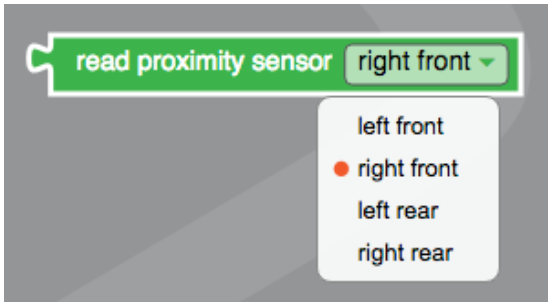
Activity:
Let the students practice converting a few numbers. Perhaps one student can convert a number to binary, the other proofs it by converting it 'back' to decimal.

Evo's Proximity Sensors

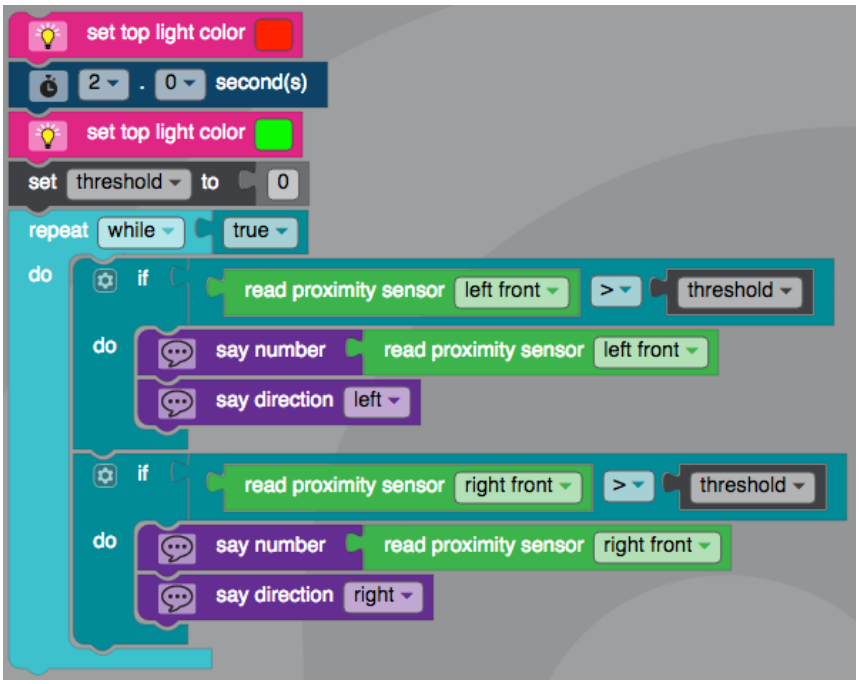


The Ozobot Evo has two proximity (infrared) sensors at the front and two at the rear. As pictured here, we will refer to them as S-LF (sensor left front), S-RF (sensor right front), S-LR (sensor left rear) and S-RR (sensor right rear). This reflects the Ozoblockly code:





Activity:
If the students aren't familiar with the sensors demonstrate some basics with a simple program such as shown below.

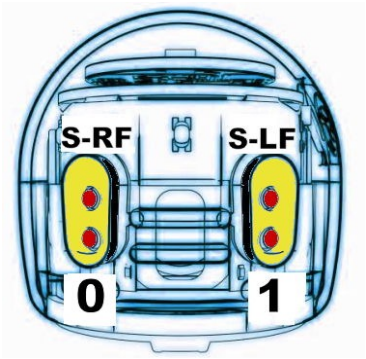


The two main takeaways are:

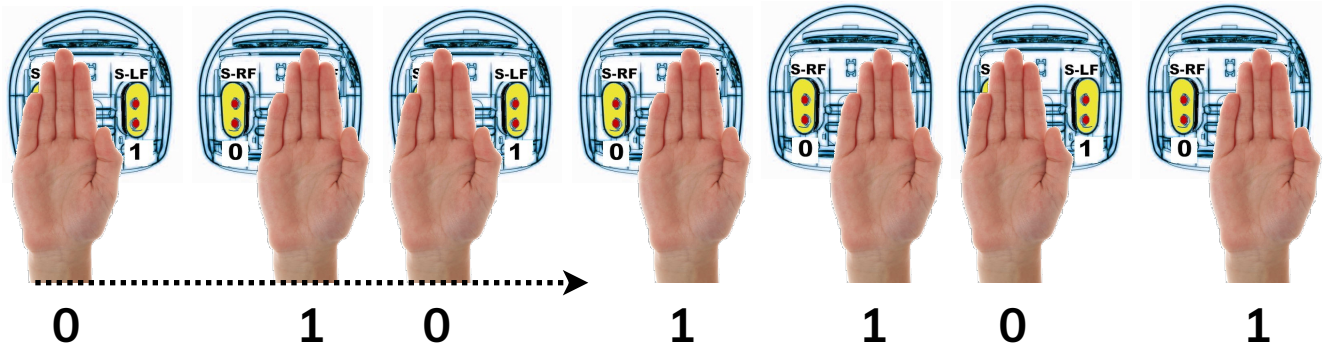
1. The sensor returns a numerical value depending on the distance of an object.
2. We can use this value as a threshold as to when we want the Evo to perform a certain action.

Entering Numbers Via The Proximity Sensors

Coming back to our "*Converting Numbers*" example from above. The result for the decimal number 90 in binary was: 1011010.



With the front of the Evo facing the user, Sensor S-RF is used to enter a '0' and sensor S-LF to enter a '1'. When the program has been started (or the function has been called), we can enter up to 7 digits for any number in the range of 0..127. The binary number is entered from right to left, or from the least significant bit to the most significant bit.



Move one hand close to the desired sensor until you hear Evo say zero or one. Remove your hand quickly and repeat until the binary sequence has been entered.

The two rear sensors can be used to terminate the input. This is useful for very small numbers, say you want to enter 5 (=101_b). So instead of entering 0000101, you can enter 101 and then moving your hand close to one of the rear sensors.

A short video clip shows how easy the process actually is: video-clip.

Activities:

1. Load the ‘standalone’ program onto any Ozobot Evo. Students practice entering binary numbers.
2. Load the ‘standalone’ program onto any Ozobot Evo. Teams of two students. One student enters a binary number (the other student doesn’t see it) and the Evo speaks the (decimal) number. Now the other student is tasked to convert the number just heard to binary. Do both student’s binary numbers match?
3. Load the ‘function’ code onto the student’s Ozoblockly editor. Students can modify the calling program and experiment.
4. Having the ‘function’ code available, students can use it for a variety of their own projects.

Extensions

Expand the function to be able to enter negative numbers. For example one rear sensor could terminate the input while the other one is used to indicate a negative number (which means bit 8 is set to 1).