MORE CONTROL NETWORKS

Logic gates are essential in the design of networks that control many devices. In this lesson, we will investigate and construct four examples of common control networks: (1) a system warning a driver if one or more of the four car doors is ajar, (2) a system that allows controlling a single light with two switches, such as those found at the top and bottom of a staircase or at each end of a long hallway, (3) a system that controls an outdoor yard light, sensing daylight and night, but allowing manual override, and (4) a controlled inverter, that allows inverting a computer word (a sequence of 0/1 bits) via a control signal.

ACTIVITY 1: Car "Door Ajar" Warning Circuit

Most cars contain a variety of warning signals on their dashboards. One common warning tells the driver if any one or more of the car doors is ajar, thus signaling a risk to the safety of passengers. One way to implement this is to have a push-button switch on each door that is ON when the door is properly closed. When the door is open or ajar, then the switch is OFF.

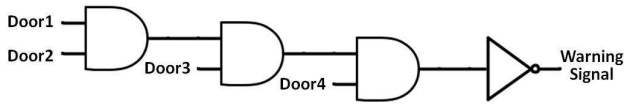
A NAND gate provides a perfect way to implement the "car door ajar" warning. This gate has a truth table as follows

A	B	A AND B	A NAND B
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

We first note from the truth table, that AND provides an ON signal only if both inputs are ON, whereas NAND provides an OFF signal only if both inputs are ON. For a 2-door car, the NAND gate would turn ON a warning signal if either or both car doors are ajar (0 or OFF). If both doors are properly closed (1 or ON), then the warning signal would be OFF. So a NAND gate is the perfect solution for this warning system.

This logic can easily be extended to such a warning signal for a 4-door car. All we need is a NAND gate with four inputs. For a 4-door car when all four doors are properly closed (1 or ON), then the warning signal would be OFF. Otherwise, when one or more doors are ajar (0 or OFF), the warning signal would be ON.

But how do we make a four-input NAND gate? The key to answering this question lies in noting that a NAND gate is the output of an AND gate that has been inverted! If we only have two-input AND gates and NOT gates available, all we need to do is connect three two-input AND gates end-to-end, and then connect a NOT gate to the output of the final AND gate in the series:



Construct this circuit with littleBits. Use buttons to represent each of the doors. When the button is pressed down (1 or ON), the door is properly closed; when the button is up (0 or OFF), the door is ajar. For the warning signal, you could use an LED or a buzzer, for example.

ACTIVITY 2: Controlling a Single Light with Two Switches

An interesting problem is to design a logic circuit with two switches and a single LED. If the LED is off, then either switch can be used to turn it on. Alternatively, if the LED is on, then either switch can be used to turn it off.

An analog circuit analogy is the problem of controlling a light in a stairwell with a switch at the top and bottom. Electricians have known and implemented designs for such switches and their corresponding circuits for decades. But here we turn our attention to a digital logic circuit design that solves this problem.

Either of two different logic gates can be used to solve this problem—and exclusive-OR gate (XOR) or an exclusive-NOR gate (XNOR).

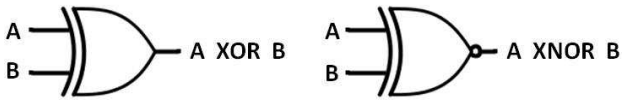
The truth table for these gates is as follows:

A	B	A XOR B	A XNOR B
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Note that the XNOR gate is simply the output of the XOR gate inverted. The XNOR gate is also sometimes known as an EQUIVALENCE gate, as the output is 1 only if both inputs are the same (equivalent).

With either the XOR or the XNOR gate, changing the state of either A or B will always flip the output. This is exactly what is needed to control a single LED with two switches.

These logic gates are symbolized as shown in the following figure:



Note the small circle at the far right of the XNOR gate symbol. A small circle is commonly used to indicate that the signal is being inverted.

There is no XNOR module available with littleBits. However, you can construct an XNOR gate with littleBits by using an XOR module whose output is connected to an inverter module!

Now you are ready to construct a logic circuit that will control a single light with two switches. Use your choice of either the XOR or the XNOR implementation of this circuit. It is suggested that you use either toggle switch modules or slide switch modules and your choice of an LED module.

ACTIVITY 3: Outdoor Light Control System

You are interested in a system that will control an outdoor light (and its built-in AC outlet) in your backyard. This control system should satisfy the following requirements:

1. Turn the light on at night and off at dawn.
2. Have a manual override switch that would allow the light to be off during the night and on during the day. *Off during the night* might be useful if you are hosting a party in your back yard and want only ambient lighting and not a glaring yard light. *On during the day* would be useful if you wanted to plug in some yard maintenance equipment such as a trimmer/edger.

You can use a light sensor module with the trigger set to sense *dark*, and the sensitivity set fairly high. With the light sensor set in this way, it will output a 1 (ON) at night when it is dark, and a 0 (OFF) during the day when it is light.

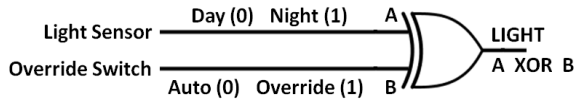
The truth table for this control system can then be set up as follows:

Light Sensor	Override Switch	LIGHT
Day (0)	Auto (0)	OFF (0)
Day (0)	Override (1)	ON (1)
Night (1)	Auto (0)	ON (1)
Night (1)	Override (1)	OFF (0)

A discussion of the truth table goes like this. The light sensor is OFF (0) during the day and ON (1) during the night. When the override switch is in auto (0 or OFF), then the light sensor determines if the light is on or off. When the override switch in ON (override), then the light is inverted from what the light sensor indicates.

You can then observe that the truth table for the LIGHT output is the exclusive OR (XOR) of the light sensor and the override switch.

The logic circuit diagram for our outdoor light control system then looks like this:



You should now be able to construct the complete circuit using littleBits. You need to keep in mind that the light sensor can be sensed by the red led on the littleBits power module, so you will want to keep the power module at least a wire's distance from the light sensor! You can test your circuit in a room in which you can easily turn the room lights on or off.

If you are interested in programming, it would also be an instructive project to design an Arduino sketch for this outdoor light control system, and then test your sketch by using a littleBits Arduino module. An earlier lesson, entitled "Toxic Waste Control System" contains details on the use and programming of the Arduino module.

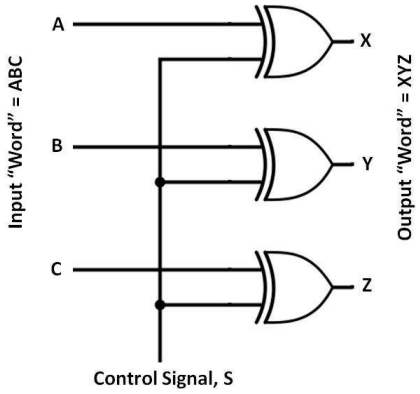
ACTIVITY 4: Designing a Controlled Inverter

Begin by considering the truth table for the XOR gate:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

First note that 0 XOR B = B, regardless of whether B is 0 or 1. Also, note that 1 XOR B = B' (B' is commonly used to designate B inverted), regardless of whether B is a logic 0 or 1. The result of these two facts is that an XOR gate can be used as a NOT gate by making one of its inputs a logic 1.

In computer hardware, XOR gates can therefore be used to invert every bit in a **word** (*i.e.,* a sequence of bits) by using one input for each of the XOR gates as a control line. The following figure shows such a circuit for a three-bit word:



When the control signal is 0, then the output word XYZ is the same as the input word. However, when the control signal is 1, then the output word is A'B'C', with the bits inverted.

You should now be able to construct a three-bit controlled inverter with littleBits modules. It is suggested that the inputs A, B, and C be represented by button modules, the control signal by a slide or toggle switch, and the outputs X, Y, and Z be represented by LED modules.