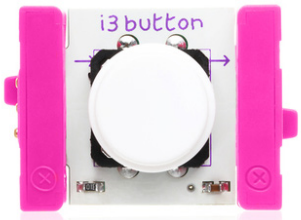
LOGIC MODULES
INTRODUCTION

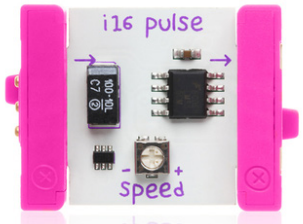
littleBits
EXPANSION ACTIVITY

With littleBits logic modules, you can program in block form. The logic modules create rules for your circuit to follow, giving you more ability to create interesting and complex interactions. In this lesson, you will learn how to combine various inputs to achieve desired output results with the help of logic gates (AND, OR, NAND, NOR, XOR).

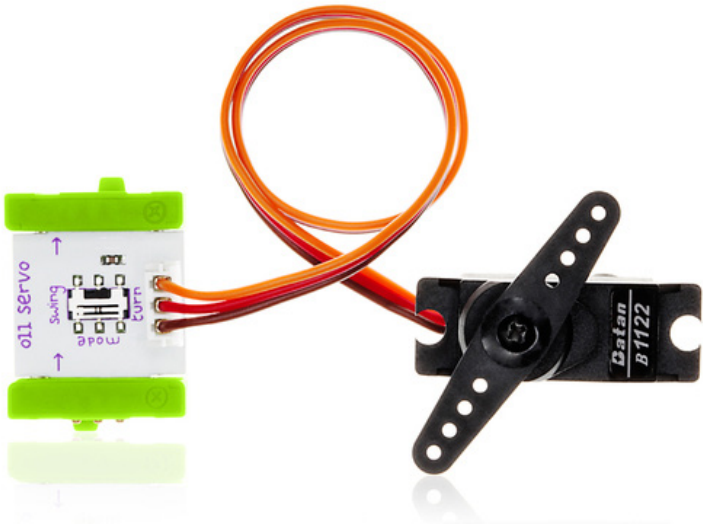
Building circuits with logic modules is practically the equivalent of physical programming, at a littleBits level. With logic, we deal with inputs and outputs. Let's break it down.

INPUT MODULES are pink. They are buttons, switches and sensors: the eyes + ears of the system. They interpret their surroundings to make things happen. Inputs tell the outputs what to do.





OUTPUT MODULES are green. These lights, sounds, and motors convey the visual, physical + audible, and just generally make stuff happen. Blink it! Buzz it! Shake it! Turn it!



LOGIC MODULES
LOGIC IN THE REAL WORLD

littleBits
EXPANSION ACTIVITY

The world is full of shades of gray and colors, but when we're looking for logic, we are checking for what can be on or off, open or closed, or true or false. These pairs, or binary relationships, can be written down into a state table, also, called a truth table.

The idea of a logic gate is that it checks what goes in and determines what goes out based on various input scenarios. In the real world, we can think of logic gates as physical gates. For example, if you have a fence, you install a gate. You can now walk to the gate, open it and then walk through. In this example you are the input into the gate and the gate outputs you into the yard. We can control the behavior of a gate, for instance, by putting a lock on it. If there is one lock on the gate you need to have one key in order to get in. If there are two different locks on the gate you need to have two different keys to get in. In the case of one lock you can say you have a key or don't have a key. So if it's true you have the key, you can open the gate. If you don't have a key it's false that you can open gate. You can write this down into a table.

KEY STATUS	LOCK STATUS
FALSE	FALSE
TRUE	TRUE

Now, in the case where there are two locks and one gate. The situation is slightly different. The only way to get through this gate and to the other side is to unlock lock1 and to unlock lock 2. Here we are working with an AND operation. We can write a table like this:

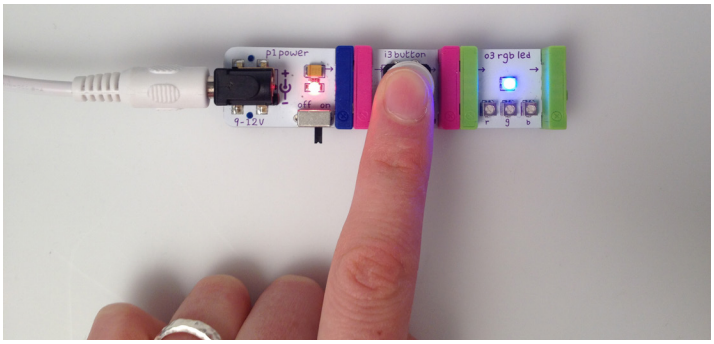
Do I have the key to lock #1?	Do I have the key to lock #2?	Can I open the gate?
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

If we had three locks, we would need all three locks to be open in order to go through the gate. There are now twice as many states for our inputs to have. If we start with one lock, we have two states, open and closed. When we have 2 locks, and because each input has two states, there are a total of 4 distinct states (2x2). Our truth table would demonstrate each of the four cases in which the inputs are triggered, or activated. With 3 inputs, we have 2x2x2 for a total of 8 states.

LOGIC MODULES
STATES

littleBits
EXPANSION ACTIVITY

If you press a button, you change the state of an input from off to on. The following output will also go from off to on. These states take some kind of action or trigger in order to change. If you lose power everything turns to off.



An activated input can be described in many ways. When we press a button before a motor module, the motor shaft will turn. In this example, energy passes through the input and can now flow on to the next module.

When it is true that energy is flowing, we can describe the status of the input as high. When the button is no longer activated (you stopped pressing it), it is false that energy is flowing through, and we can describe the status of the input as low. It is also common to represent true, and false as a "1", and a "0". The trigger for state change is the presence or lack of energy.

This is the foundation for boolean logic. When an action can be either true or false, it has two states. When an input can be described as having two states, they have a binary relationship. With binary numbers, we can describe false by calling it 0, and true by calling it 1.

A key challenge in working with logic is to ask questions that create states that can be paired together. For instance we can ask the question, "is a building tall?", but we can't ask how high the building is.

STATE A	STATE B
1	0
TRUE	FALSE
HIGH	LOW
ON	OFF
OPEN	CLOSED
ACTIVE	INACTIVE
UP	DOWN

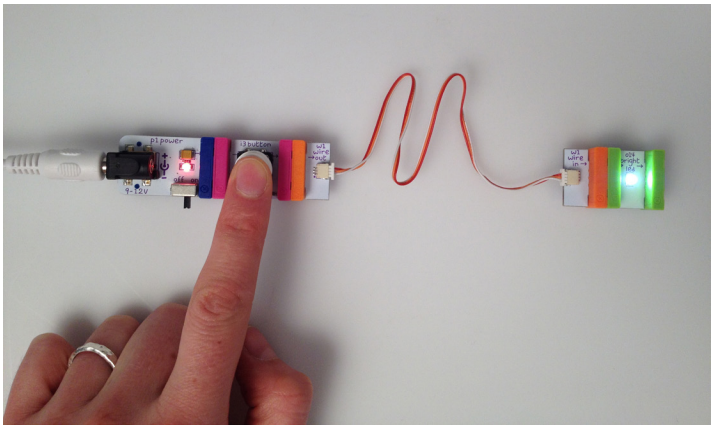
LOGIC MODULES
SERIES AND PARALLEL CIRCUITS

littleBits
EXPANSION ACTIVITY

littleBits circuits can be connected in a couple different ways, series and parallel. For instance, if we linked together a chain of buttons (or any other inputs) along a single path, we could describe them as being linked in ***SERIES***. This means that all the inputs have to be activated, pushed down, or switched, at the same time in order for an output to be turned on. This is the same theory behind AND logic.

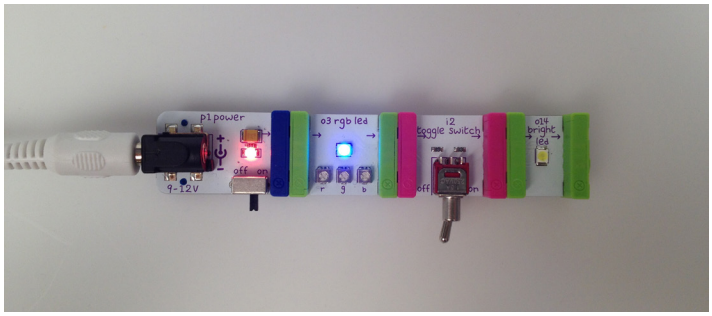
When we connect an output module to an input module, and activate the input, an electronic signal (energy) from the power module passes through the input and crosses over into the output.

Wire modules can be used to extend this connection as they do not interfere, but simply carry this signal from one module to the next. The signal that goes into a wire is the same that goes out. Wires are important for organizing and configuring circuits to fit your project needs.

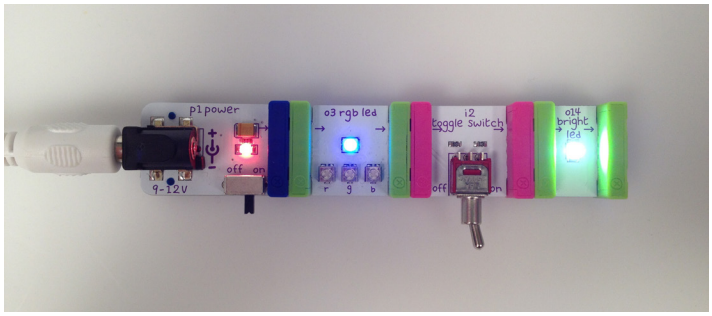


WIRES CARRY SIGNAL FROM ONE MODULE TO THE NEXT

In the wire module family, you can also find branches, forks, and splits. These modules are similar to the wire in that they don't interfere with the electrical signal, and carry it on to the next module. You can use them to branch/split your circuit in order to make parallel circuits, which are needed for some logic operations.



WIRED IN SERIES WITH THE SWITCH ON. THE LED BIT IS OFF.



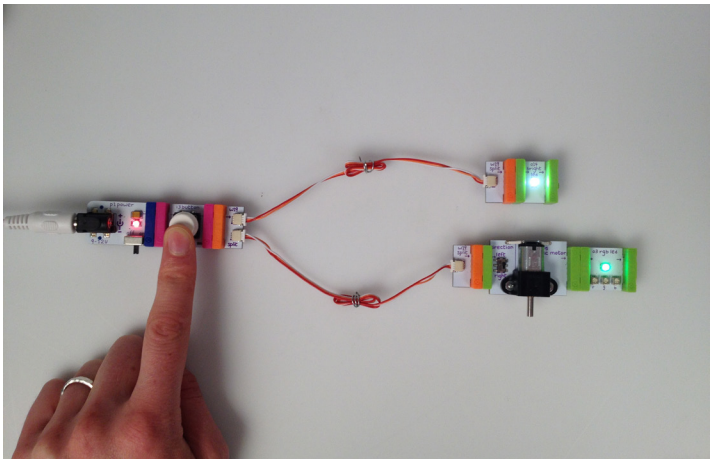
WIRED IN SERIES WITH THE SWITCH ON. THE LED BIT IS ON.

LOGIC MODULES
SERIES AND PARALLEL CIRCUITS

littleBits
EXPANSION ACTIVITY

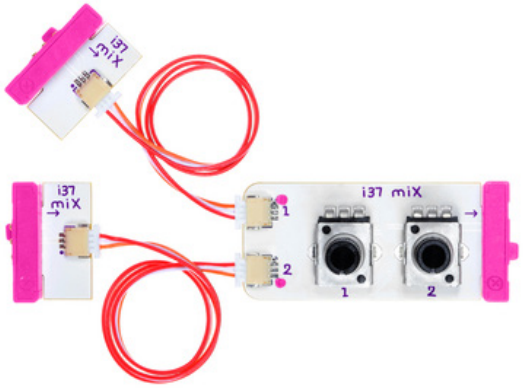
PARALLEL CIRCUITS, utilizing forks, branches, and splits, send signals from the power module along multiple, parallel paths. Place an input before the split, branch, or fork to activate multiple outputs at once. Place two inputs after the split to control independent outputs.

When working with logic modules, the signal needs to be sent along parallel paths to multiple inputs. These inputs feed into logic gates and determine the output.

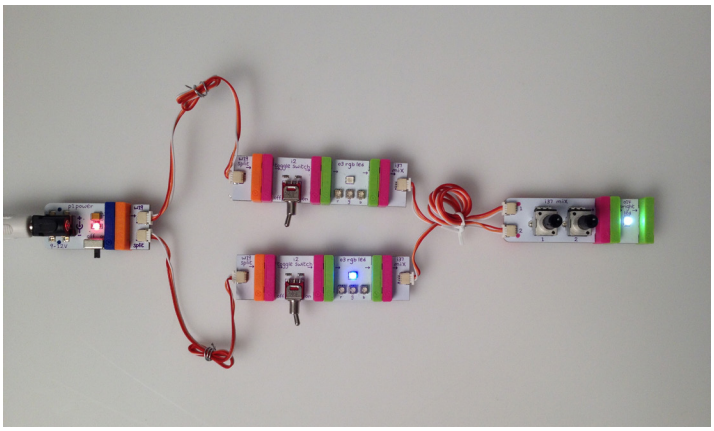


PARALLEL CIRCUIT WITH SPLIT MODULE

Occasionally, you may want to combine two parallel inputs into one output. You can do this with the mix module.



The example below mimics the behavior of an OR module by wiring switches in parallel.

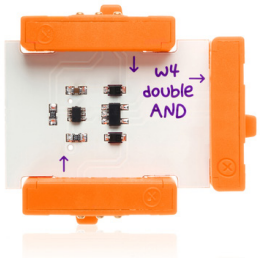


A MIX MODULE MIMICS THE BEHAVIOR OF AN OR GATE

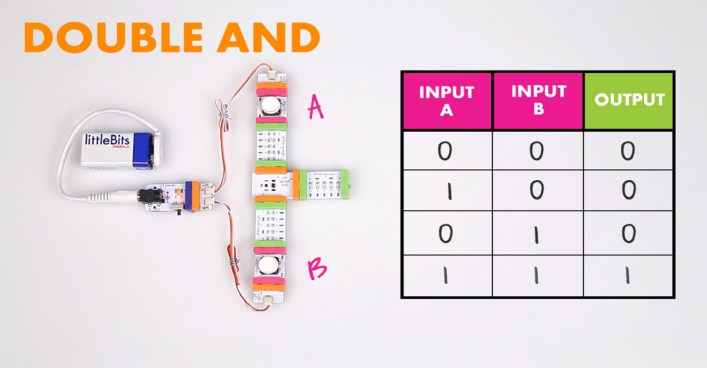
By wiring littleBits together series and parallel, it is possible to create circuits that use AND and OR logic without the need for a specific logic module. However, when you want to make more complex behaviors and automated actions, you will want to use logic modules.

LOGIC MODULES
AND GATE

littleBits
EXPANSION ACTIVITY



The AND module is a logic gate with two inputs and one output. In order for the AND gate to output an on signal, both inputs need to be on at the same time. When this happens, you are able to activate any output module like an LED, a DC motor, or a buzzer. The AND module is a good option for projects in which you want two actions to trigger another.



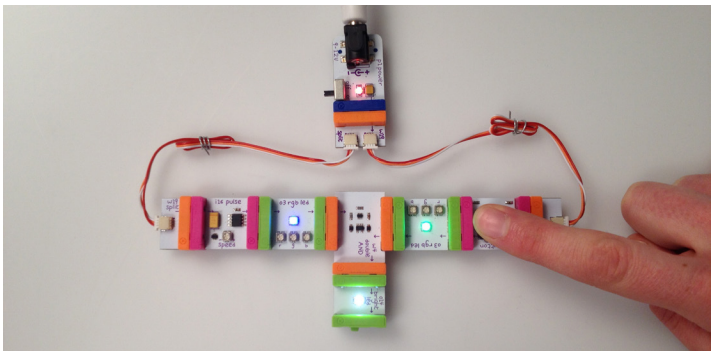
INPUT A	INPUT B	OUTPUT
0	0	0
1	0	0
0	1	0
1	1	1

TO SET UP THIS CIRCUIT, YOU NEED:
1 AND module
1 power
1 split (can also use branch or fork with wires)
2 input modules
1 output module
3 LED modules (optional)

This circuit acts similarly to a circuit in series with two inputs and one output. In both cases, the two inputs need to be on in order to turn the output on.

REAL WORLD EXAMPLE:
The AND function can be demonstrated by thinking about a door with two locks. The only way to get through this door is to unlock lock1 AND to unlock lock 2. Otherwise you will be stuck outside.

ACTIVITY: CAN YOU KEEP A BEAT?
Tap along with a metronome. This game tests if you can keep a beat. First, you need to make a metronome. A metronome is any device that produces pulses or beats at a constant speed. Many musicians use metronomes to maintain tempo when playing instruments. Make the following circuit:

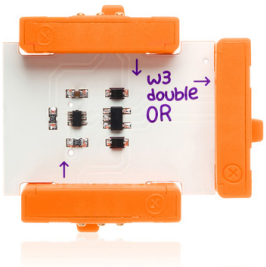


YOU NEED:
1 power
1 split
1 button/switch
3 LEDs (optional)
1 pulse
1 AND
1 output module

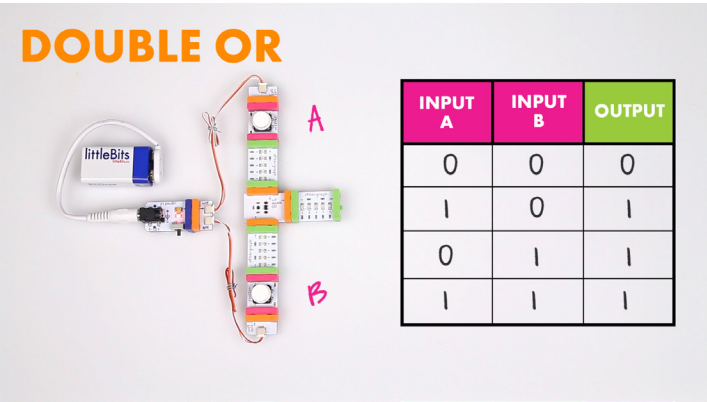
Set the pulse to whatever speed you like. The LED that follows the pulse will now blink at a regular rate. Press the button (your second input) to beat of the pulse. If you are pressing perfectly in time with the pulse, the output will be activated during the intervals in which both inputs are on. If you change the speed of the pulse, you change the rate of the metronome. See how fast or slow you can go to keep in time.

LOGIC MODULES
OR GATE

littleBits
EXPANSION ACTIVITY



The OR module is a logic gate with two inputs and one output. In order for the OR gate to output an on signal, the first input, the second input, or both inputs need to be on. When both inputs are off (inactive), the output is off. The OR is good option for projects where you want to detect two inputs but don't care which input is activated.



The OR gate has the same logic as two having inputs wired in parallel.

TO SET UP THIS CIRCUIT, YOU NEED:
1 OR module
1 power
1 split (can also use branch or fork with wires)
2 input modules
1 output module
3 LED modules (optional)

REAL WORLD EXAMPLE:
Picture a house that has a doorbell at the front door and a doorbell at the back door. Both doorbells are connected to the same buzzer inside the house. This buzzer will ring when EITHER OR BOTH of the doorbells are pressed.

ACTIVITY: AUTOMATIC NIGHT LIGHT
Activate a night light when a person enters a room using an OR module, a sound trigger, and a motion trigger. As long as there is motion, sound, or both, the light will stay on. When the person leaves the room or falls asleep, there is neither sound nor movement, so the light will turn off automatically. To check this we can use a truth table.

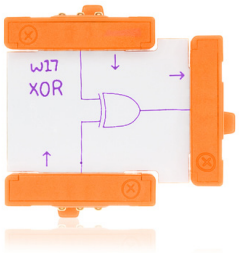
YOU NEED:
1 power
1 split
1 motion trigger
1 sound trigger
1 OR
1 bright LED (more are optional)

motion trigger (INPUT A)	sound trigger (INPUT B)	LED status
FALSE	FALSE	OFF
FALSE	TRUE	ON
TRUE	FALSE	ON
TRUE	TRUE	ON

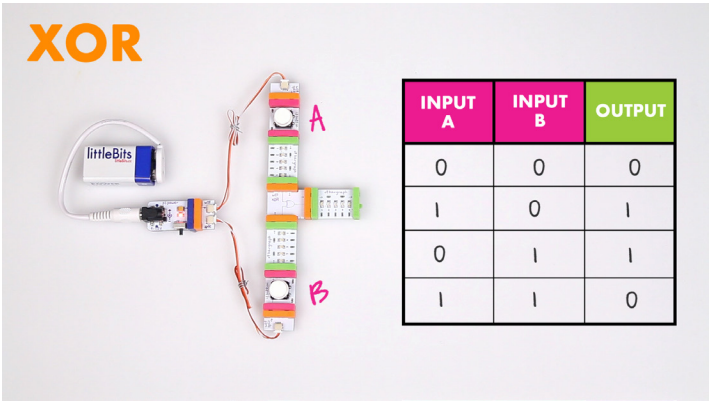
LOGIC MODULES

XOR GATE

littleBits
EXPANSION ACTIVITY



The XOR module is a logic gate with two inputs and one output. In order for the XOR gate to output an on signal, either one of its inputs, but not both, are on. The XOR is called the exclusive OR because it is similar to the OR gate, but only outputs a signal when one input is on at a time. The XOR is great for projects in which the activation requires input alternation.



INPUT A	INPUT B	OUTPUT
0	0	0
1	0	1
0	1	1
1	1	0

TO SET UP THIS CIRCUIT, YOU NEED:
1 XOR module
1 power
1 split (can also use branch or fork with wires)
2 input modules
1 output module
3 LED modules (optional)

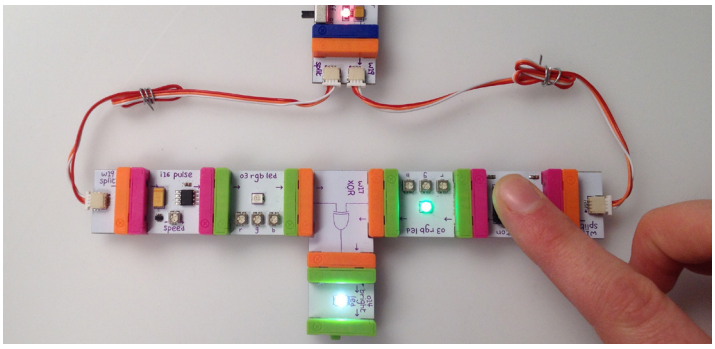
The XOR module cannot be created by only wiring inputs in parallel or series to an output. Only the AND, and OR module mimic the behavior of connecting inputs in parallel or series.

The XOR module contains the logic used in adding binary numbers. Adding 0 + 0 equals 1. Adding 1 + 0 or 0 + 1, equals 1. Using the XOR for 1 + 1 results in an output of 0. Because of this, we need more than one XOR to add binary nmbers. Later on, we will explore how to create a half adder, and full adder with XOR gates.

REAL WORLD EXAMPLE:
The XOR is great for projects in which the activation requires alternation. Because the output is only true when one input is true, this circuit can be used to make sure that only one event happens at a time. The XOR also works well in scenarios in which you want one button to start an action, and the other button to stop it.

ACTIVITY: CAN YOU FIND THE UP BEAT?
Tap opposite the metronome. This game tests if you can keep an up beat. When you count beats in music you count the downbeat. Half way between two down beats is what is called the upbeat. When you play in time with the up beat, this is called syncopation.

First, you need to make a metronome. A metronome is any device that produces pulses or beats at a constant speed. Many musicians use metronomes to maintain tempo when playing instruments. Make the following circuit:



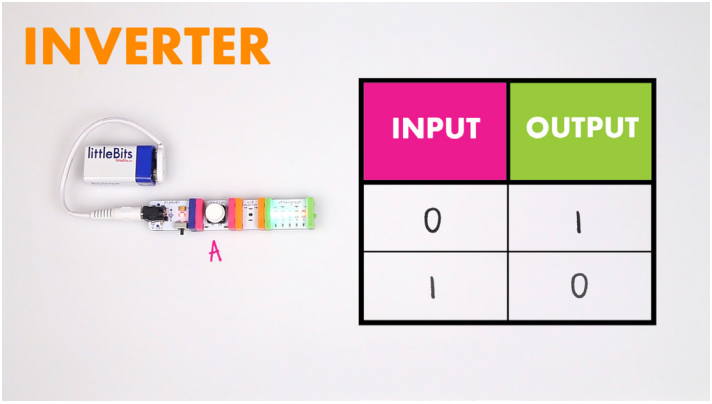
Set the pulse to whatever speed you like. The LED that follows the pulse will now blink at a regular rate. Press the button (your second input) in between the beat of the pulse. You will know that you are keeping the upbeat when your output light remains on and does not blink out. If you change the speed of the pulse, you change the rate of the metronome. See how fast or slow you can go to keep a syncopated rhythm.

YOU NEED:
1 power,1 split, 1 button/switch, 3 LEDs (optional), 1 pulse, 1 AND, and 1 output module.

LOGIC MODULES
NOT

littleBits
EXPANSION ACTIVITY

The Inverter (or NOT) module has one input and one output. With this module, the state of the input is inverted to its opposite state. If the input is on, then the output is off. If the input is off, then the output is on. Try placing the inverter in between two lights and after a button. Pressing the button will make the LEDs blink back and forth, like the lights on top of a police car! You can also try chaining two inverters in series. You will see that the output is true, due to the double negative.



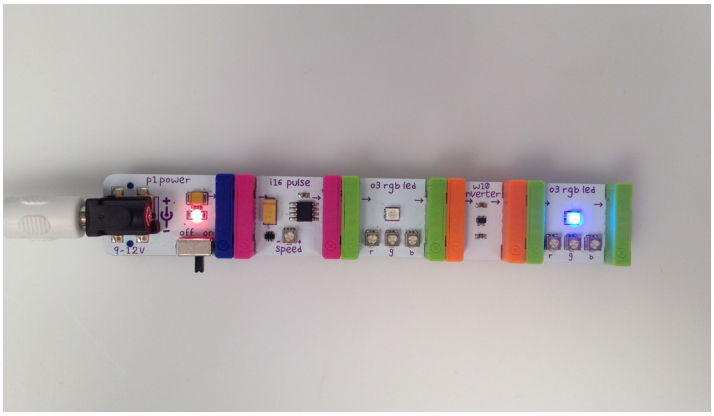
TO SET UP THIS CIRCUIT, YOU NEED:
1 power
2 LEDs
1 input module
1 inverter

Inverters can be used in combination with logic gates to essentially change how the logic works. If you place the inverter after the output of a gate, it will change the logic completely. If you place the inverter before one input, it changes the logic going into the gate from that input. Certain combinations of logic gates and inverters create new gates with different functions. For example, when the AND module is combined with a NOT gate, we get the NAND gate. When the OR is combined with the NOT module with get the NOR gate. These gates are explained in more detail in the NAND, and NOR section.

REAL WORLD EXAMPLE:
The lights on top of a police car flash back and forth. A pulse sends a signal to both of the lights, but they flash opposite one another due to the inverter.

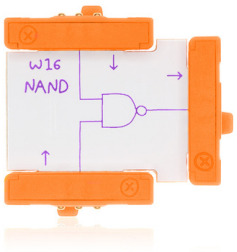
ACTIVITY: PLAY WITH BLINKING LEDS AND THE NOT GATE
Set up a chain of LEDs after a pulse module. Place an inverter anywhere along the chain and see how the lights alternate. Often times, when setting up logic gates, we will use LEDs to help us to visually decode what is happening in the circuit. With LEDs, you can see the state of the input before it goes into the NOT gate.

YOU NEED:
1 power
1 pulse
1 inverter (or more)
2 or more output modules

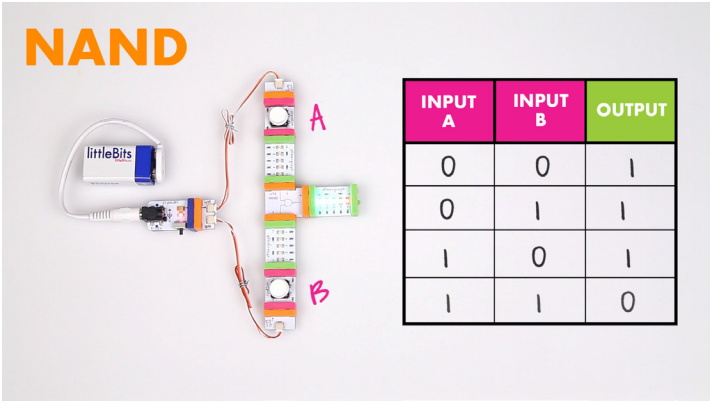


LOGIC MODULES
NAND GATE

littleBits
EXPANSION ACTIVITY



The NAND module is a logic gate with two inputs and one output. NAND is short for NOT AND which is the combination of an AND gate where the output is inverted. The NAND only outputs an off signal when both of the inputs are on, or true. If both inputs are off, the output signal will be on.



INPUT A	INPUT B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

TO SET UP THIS CIRCUIT, YOU NEED:
1 NAND module
1 power
1 split (can also use branch or fork with wires)
2 input modules
1 output module
3 LED modules (optional)

Another way to make a NAND from other logic modules is to invert both inputs going into an AND gate. Making custom gates takes more modules, which is why we have the NAND gate. It makes circuit building more efficient.

The NAND gate is also known as a universal gate. This means that with careful arrangement of inputs and outputs to and from the NAND gate, any other logic can be created. The other universal logic gate is the NOR gate.

REAL WORLD EXAMPLE:
Think of a car with two doors. The doors act as inputs to the NAND gate. When a door is closed, that input is on. Therefore the interior light shines if the either or both of the doors are open, or not activated. The light goes off when both doors are closed.

ACTIVITY: PROTECT A PRICELESS OBJECT
You are protecting an object inside a box. You want to make sure that no one opens the box and that nobody takes it from a certain location. Make a circuit to protect your object.

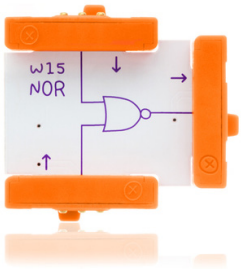
YOU WILL NEED:
1 NAND module
1 power
1 split
1 light sensor
1 roller switch
1 buzzer
2 wires

Place a roller switch inside the box and a light sensor (set on dark mode) underneath the box. When the box is closed, the roller switch is activated. The light sensor is also activated when the box sits on top of it because it shields out the light. If someone tries to open the box, they will deactivate the roller switch, setting of the alarm. The same goes for removing the box from its specified location on top of the light sensor. The alarm will also sound if both inputs are off.

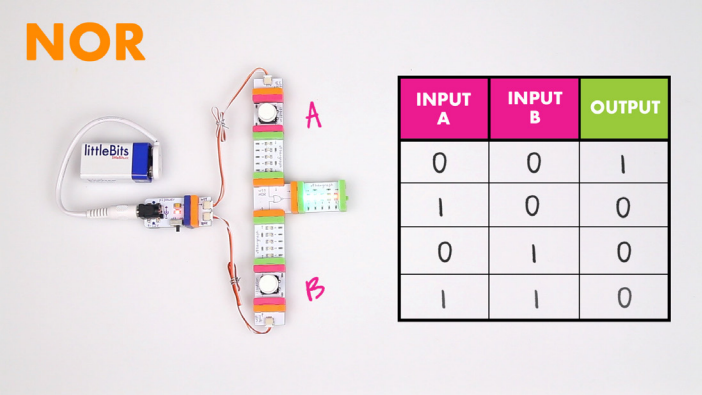
LOGIC MODULES

NOR GATE

littleBits
EXPANSION ACTIVITY



The NOR module is a logic gate with two inputs and one output. NOR stands for NOT OR, meaning the output of the OR module is inverted. With the NOR module, the output is on when both inputs are off, or false. If either or both inputs are on, the output will be off, or false. The NOR gate is good for projects in which you want the output to be on unless one or both of its inputs are triggered.



TO SET UP THIS CIRCUIT, YOU NEED:
1 NOR module
1 power
1 split (can also use branch or fork with wires)
2 input modules
1 output module
3 LED modules (optional)

REAL WORLD EXAMPLE:
A washing machine has two sensors. One to check if the washing machine lid is open and the other to sense whether the water in the washing tub is not filled to minimum level. If either or both of these inputs are activated, the washing machine will stop.

ACTIVITY: MONITOR YOUR PET'S WATER
You have two water bowls for your pet, and you want to make sure that at least one bowl is always full. Make a circuit that notifies you if both bowls are empty.

YOU WILL NEED:
1 NOR
2 pressure sensors
1 power
1 split
1 output (buzzer)
1 wire

Place a pressure sensor under each bowl. Note: you will want to put a sheet of plastic wrap or wax paper in between the bowls and your circuit so as not to get your modules wet. The weight of the water in each of the bowls will activate the pressure sensors. As long as one of the bowls is full, the output of your circuit will be off. However, if both bowls are empty, both pressure sensors will be off, and you will be notified by light, sound, or whatever you choose your output to be.