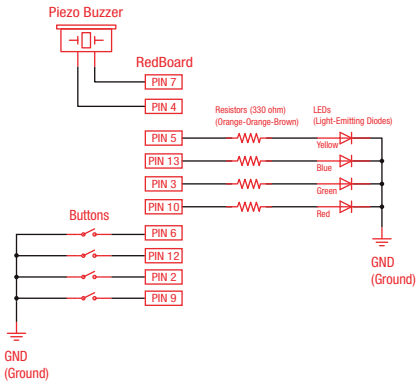
CIRCUIT #16

16

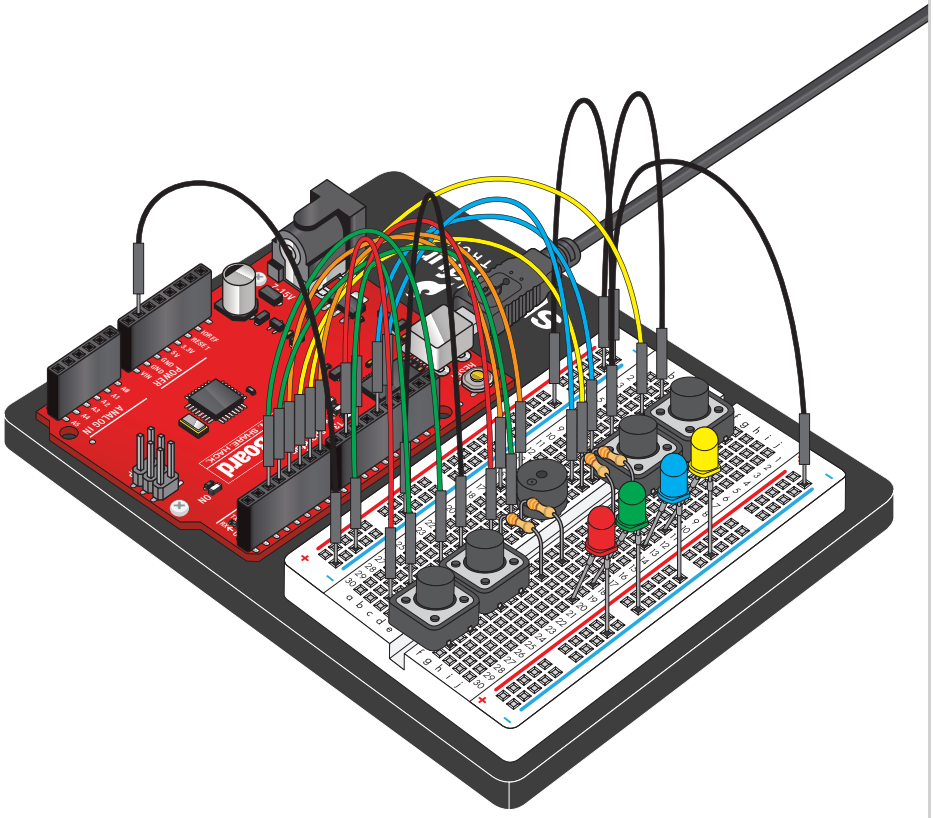
Simon Says

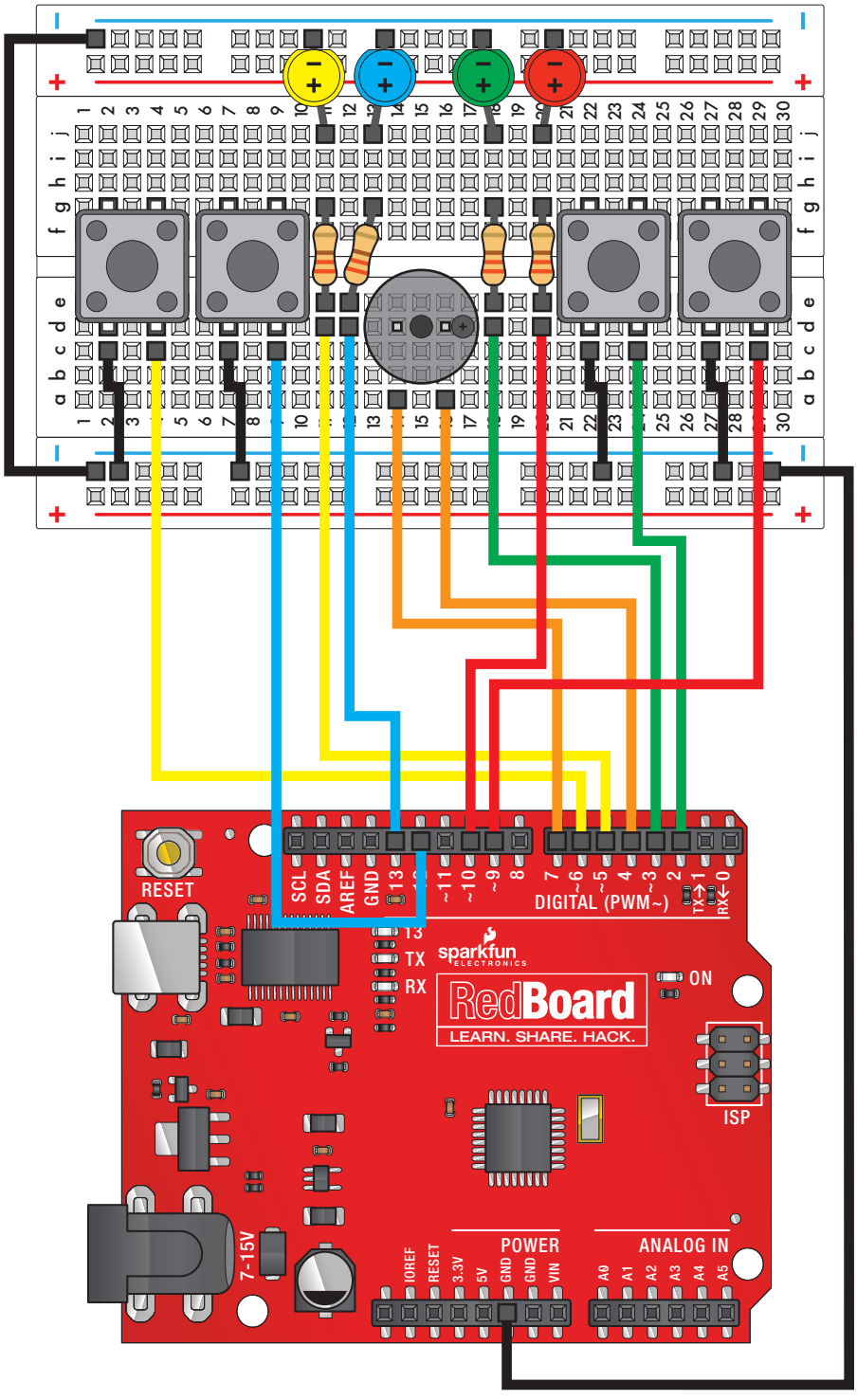


Now that we've learned all the basics behind the components in the SIK, let's put them together and have some fun. This circuit will show you how to create your own Simon Says game. Using some LEDs, some buttons, a buzzer and some resistors, you can create this and other exciting games with your SIK.

PARTS:

330Ω Resistor	LED	Push Button	Piezo Element	Wire
x4	x4	x4	x1	x16





Circuit 16: Simon Says

Component:	Image Reference:		
330Ω Resistor			e11 - g11
330Ω Resistor			e12 - g13
330Ω Resistor			e18 - g18
330Ω Resistor			e20 - g20
LED (5mm) ⚠			j11 - (+ -)
LED (5mm) ⚠			j12 - (+ -)
LED (5mm) ⚠			j18 - (+ -)
LED (5mm) ⚠			j20 - (+ -)
Push Button			d2 g2 / d4 g4
Push Button			d7 g7 / d9 g9
Push Button			d22 g22 / d24 g24
Push Button			d27 g27 / d29 g29
Piezo Element ⚠			a16 a14 (+ -)

Component:	Image Reference:		
Jumper Wire			c2 - -
Jumper Wire		Pin 6	c4
Jumper Wire			c7 - -
Jumper Wire		Pin 12	c9
Jumper Wire		Pin 5	d11
Jumper Wire		Pin 13	d12
Jumper Wire		Pin 5	d11
Jumper Wire		Pin 13	d12
Jumper Wire		Pin 3	d18
Jumper Wire		Pin 10	d20
Jumper Wire			c22 - -
Jumper Wire		Pin 2	c24
Jumper Wire			c27 - -
Jumper Wire		Pin 9	c29
Jumper Wire			- - -
Jumper Wire		GND	-

16

Arduino Code:

Open Arduino IDE // File > Examples > SIK Guide > Circuit #16

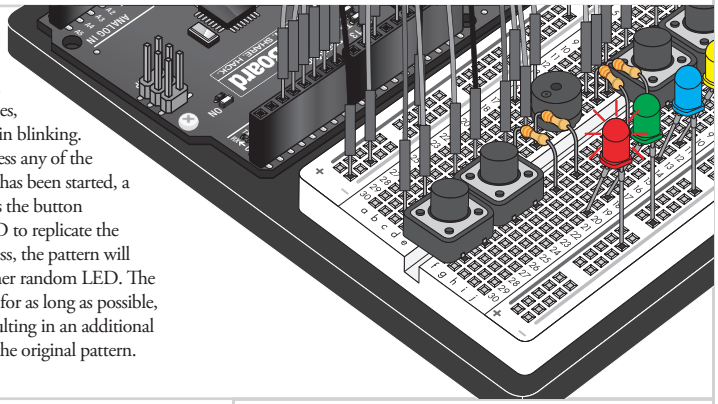
Code to Note:

#define ⇨ The #define statement is used to create constants in your code. Constants are variables that will likely only have one value during the lifespan of your code. Thus, you can assign constants a value, and then use them throughout your code wherever you need them. Then, if you need to change that value, you only have to change one line instead of going through all the code to find every instance of that variable.

byte ⇨ Bytes are another variable type. In the world of computing, a byte is a chunk of space that contains 8 bits, and a bit is a single binary value. Binary is another way of counting and uses only 1's and 0's. So a byte can hold all 1's: 11111111, all 0's: 00000000, or a combination of the two: 10010110.

What You Should See:

With the circuit complete, plug the Arduino in to a power source. Once powered, the buzzer will beep a few times, and all four LEDs should begin blinking. The game begins once you press any of the four buttons. Once the game has been started, a random LED will blink. Press the button associated with that color LED to replicate the pattern. With a successful guess, the pattern will repeat, this time adding another random LED. The player is to follow the pattern for as long as possible, with each successful guess resulting in an additional layer of complexity added to the original pattern.



Troubleshooting:

Only half the circuit works
If only half of your circuit is working, make sure you added the additional wire from one ground rail to the other. Remember that breadboards have two power rails on each side and that these can be connected, or bussed, together to provide the power to both sides of the same circuit.

No sound
Once the buzzer is in the breadboard, it's hard to see the legs and which row they are connected to. If you aren't hearing any sound, make sure your wires are on the same row as the buzzer legs.

Game is not working
If everything starts up ok, but you're having trouble when it comes time to play the game, you may have a button or two misplaced. Pay close attention to which pin is connected to each button, as it matters which button is pressed when a particular color lights up.

Real World Application:

Toys and games, such as the original Simon from Milton Bradley, have relied on electronics to provide fun and entertainment to children across the world.