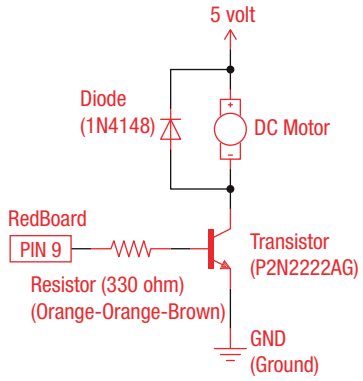
CIRCUIT #12

12

Spinning a Motor

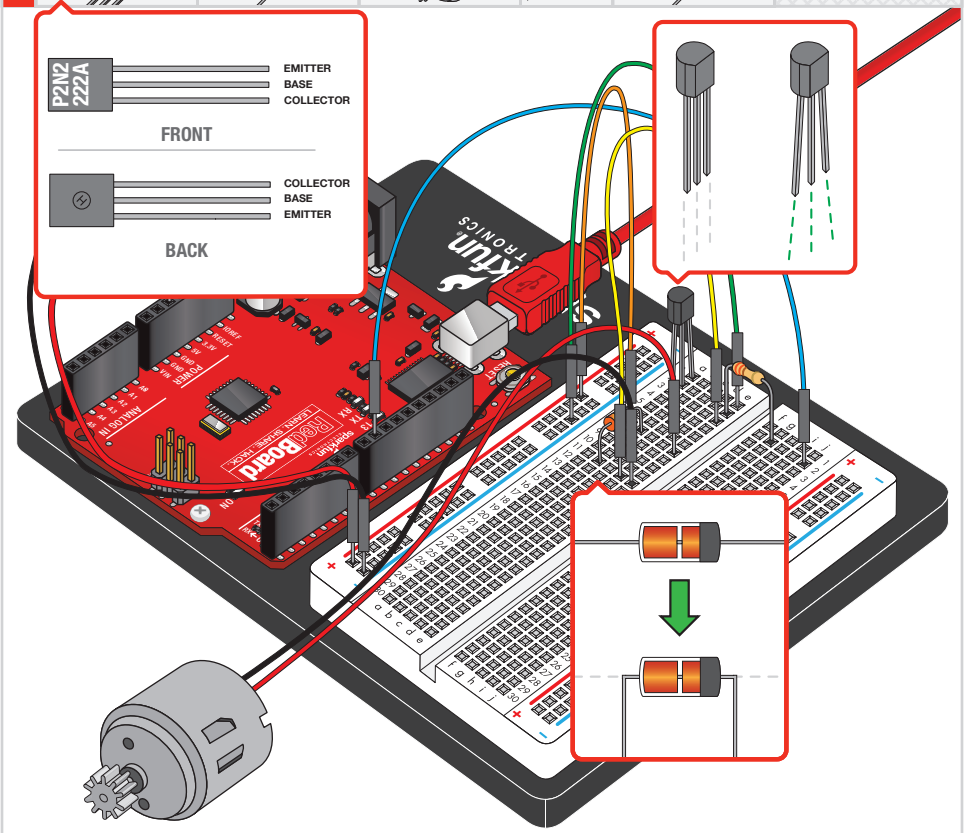


Remember before when you played around with a servo motor? Now we are going to tackle spinning a motor. This requires the use of a transistor, which can switch a larger amount of current than the RedBoard can. When using a transistor, you just need to make sure its maximum specs are high enough for your use. The transistor we are using for this circuit is rated at 40V max and 200 milliamps max – perfect for our toy motor! When the motor is spinning and suddenly turned off, the magnetic field inside it collapses, generating a voltage spike. This can damage the transistor. To prevent this, we use a "flyback diode", which diverts the voltage spike around the transistor.

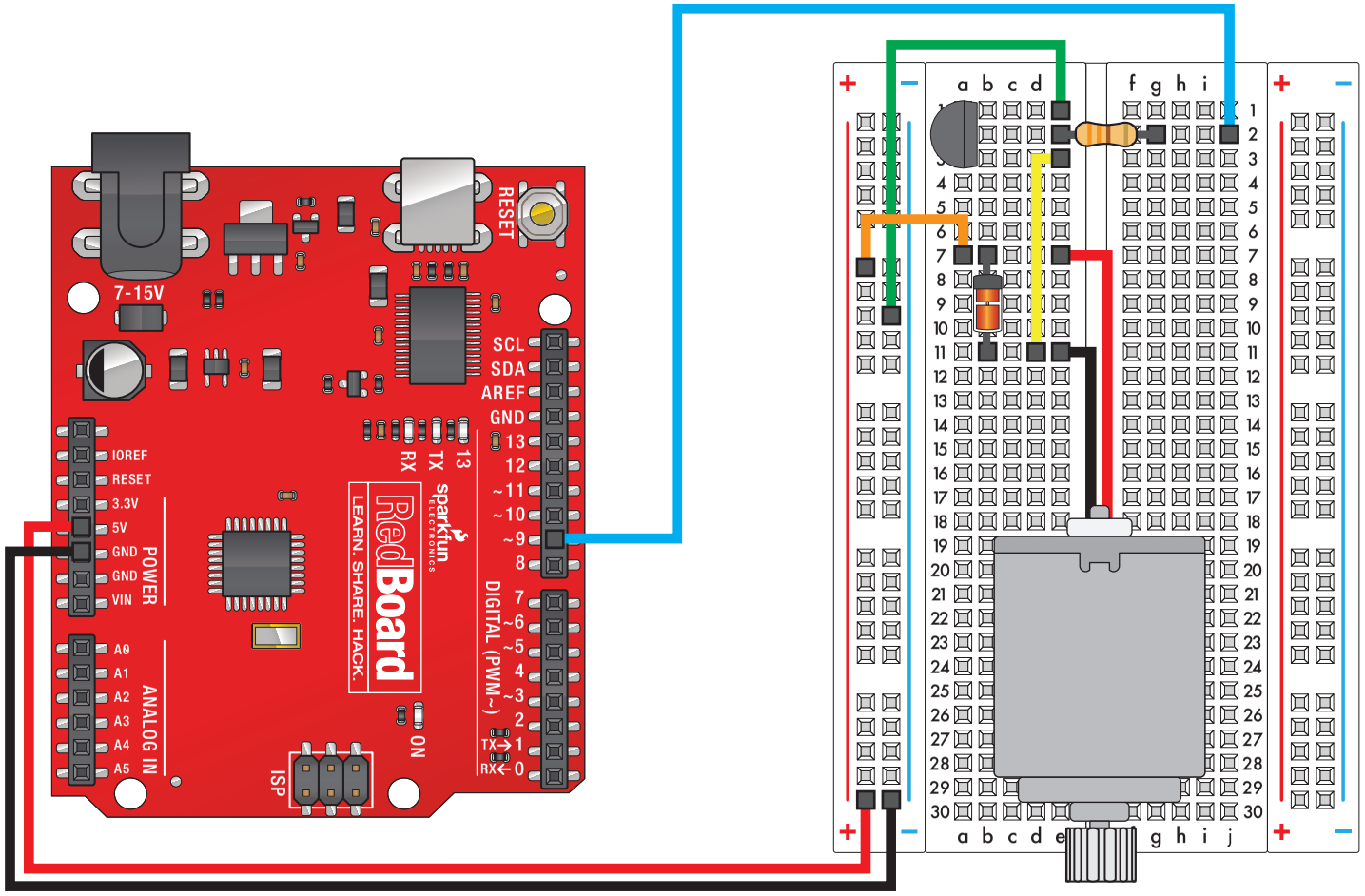
! When you're building the circuit be careful not to mix up the transistor and the temperature sensor, they're almost identical. Look for "P2N2222A" on the body of the transistor.

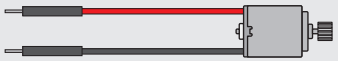
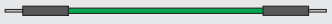
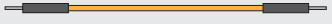
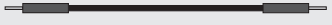
PARTS:

Transistor P2N2222AG	Diode 1N4148	DC Motor	Wire	330Ω Resistor
x1	x1	x1	x6	x1



Circuit 12 : Spinning a Motor



Component:	Image Reference:		
Transistor P2N2222AG ⚠			a1 - a2 - a3 E B C
Diode 1N4148 ⚠			b7 - b11
DC Motor			e7 - e11
330Ω Resistor			e2 - g2
Jumper Wire			e1 - -
Jumper Wire		Pin 9	j2
Jumper Wire			a7 - +
Jumper Wire			e3 - d11
Jumper Wire		5V	+
Jumper Wire		GND	-

Putting it all together:

At this point you're probably starting to get your own ideas for circuits that do fun things, or help solve a real problem. Excellent! Here are some tips on programming in general.

Most of the sketches you write will be a loop with some or all of these steps:

1. Perform some sort of input
2. Make some calculations or decisions
3. Perform some sort of output
4. Repeat! (Or not!)

We've already shown you how to use a bunch of different input sensors and output devices (and we still have a few more to go). Feel free to make use of the examples in your own sketches - this is the whole idea behind the "Open Source" movement.

It's usually pretty easy to pull pieces of different sketches together, just open them in two windows, and copy and paste between them. This is one of the reasons we've been promoting "good programming habits". Things like using constants for pin numbers, and breaking your sketch into functions, make it much easier to re-use your code in new sketches. For example, if you pull in two pieces of code that use the same pin, you can easily change one of the constants to a new pin. (Don't forget that not all of the pins support **analogWrite()**; the compatible pins are marked on your board.)

If you need help, there are internet forums where you can ask questions. Try Arduino's forum at **arduino.cc/forum**, and SparkFun's at **forum.sparkfun.com**. When you're ready to move to more advanced topics, take a look at Arduino's tutorials page at **arduino.cc/en/tutorial**. Many of SparkFun's more advanced products were programmed with Arduino, (allowing you to easily modify them), or have Arduino examples for them. See our product pages for info.

Finally, when you create something really cool, consider sharing it with the world so that others can learn from your genius. Be sure to let us know on **https://www.sparkfun.com/project_calls** so we can put it on our home page!

12 Arduino Code:

Open Arduino IDE // File > Examples > SIK Guide > Circuit # 12

Code to Note:

`while (Serial.available() > 0)` ⇨

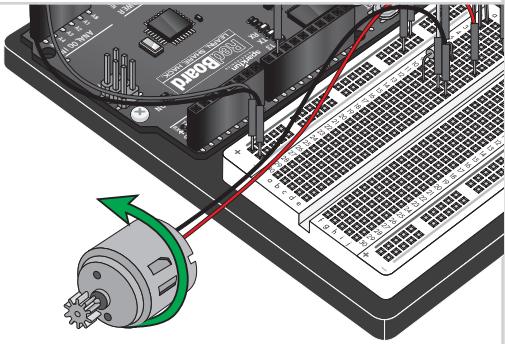
The RedBoard's serial port can be used to receive as well as send data. Because data could arrive at any time, the RedBoard stores, or "buffers" data coming into the port until you're ready to use it. The Serial.available() command returns the number of characters that the port has received, but haven't been used by your sketch yet. Zero means no data has arrived.

`speed = Serial.parseInt();` ⇨

If the port has data waiting for you, there are a number of ways for you to use it. Since we're typing numbers into the port, we can use the handy Serial.parseInt() command to extract, or "parse" integer numbers from the characters it's received. If you type "1" "0" "0" to the port, this function will return the number 100.

What You Should See:

The DC Motor should spin if you have assembled the circuit's components correctly, and also verified/uploaded the correct code. If your circuit is not working check the troubleshooting section below.



Troubleshooting:

Motor Not Spinning
If you sourced your own transistor, double check with the data sheet that the pinout is compatible with a P2N2222AG (many are reversed).

Still No Luck
If you sourced your own motor, double check that it will work with 5 volts and that it does not draw too much power.

Still Not Working
Sometimes the RedBoard will disconnect from the computer. Try un-plugging and then re-plugging it into your USB port.

Real World Application:

Radio Controlled(RC) cars use Direct Current(DC) motors to turn the wheels for propulsion.