

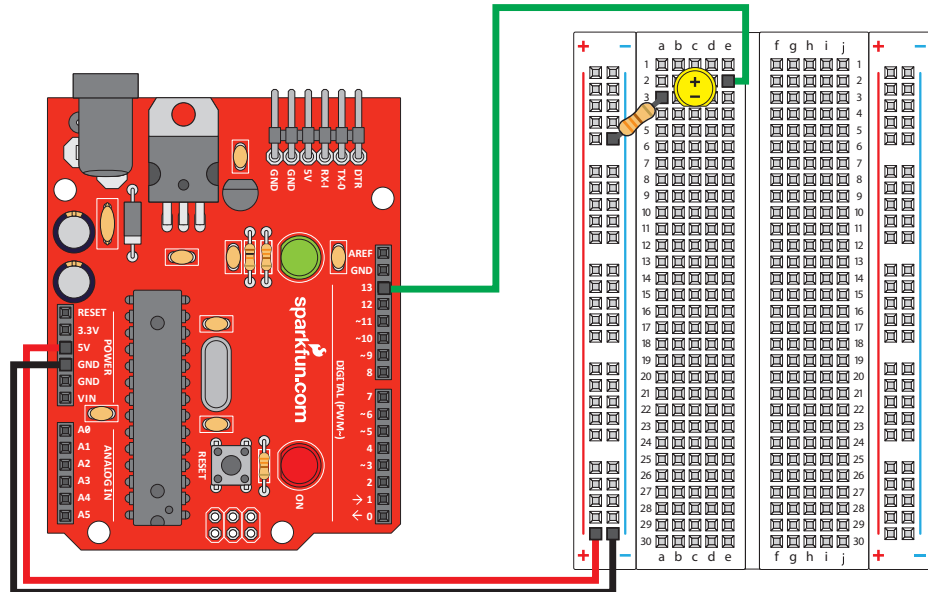
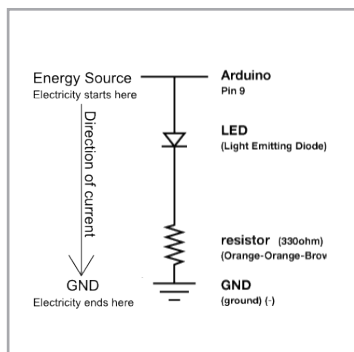
CHAPTER 2

How the Circuits Work

Circuit #1: Blinking an LED

Name:

Date:



Explanation:

This circuit takes electricity from digital Pin # 9 on the RedBoard. Pin # 9 on the RedBoard has Pulse Width Modulation capability allowing the user to change the brightness of the LED when using analogWrite. The LED is connected to the circuit so electricity enters through the anode (+, or longer wire) and exits through the cathode (-, or shorter wire). The resistor dissipates current so the LED does not draw current above the maximum rating and burn out. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pin # 9: Power source, PWM (if code uses analogWrite) or digital (if code uses digitalWrite) output from Arduino board.

LED: As in other diodes, current flows easily from the + side, or anode (longer wire), to the - side, or cathode (shorter wire), but not in the reverse direction. Also lights up!

330 Ohm Resistor: A resistor resists the current flowing through the circuit. In this circuit the resistor reduces the current so the LED does not burn out.

Gnd: Ground

Code:

```
int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH); //LED on
  delay(1000); // wait second
  digitalWrite(ledPin, LOW); //LED off
  delay(1000); // wait second
}
```

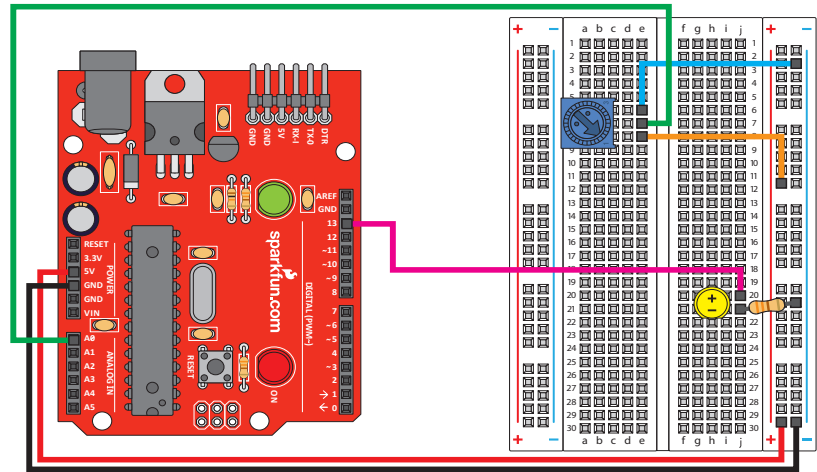
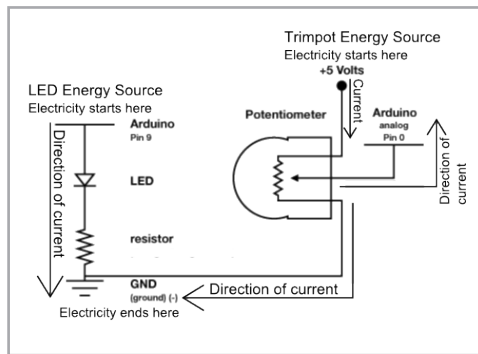
or for PWM the output loop could read :

```
int ledPin = 11;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  analogWrite(ledPin, 255); // LED on
  delay(1000); // wait second
  analogWrite(ledPin, 0); // LED off
  delay(1000); // wait second
}
```

Circuit #2: Potentiometer



Explanation:

This circuit is actually two different circuits. One circuit for the potentiometer and another for the LED. See 'How the Circuits Work' Circuit 1 for an explanation of the LED circuit. The potentiometer circuit gets electricity from the 5V on the Arduino. The electricity passes through the potentiometer and sends a signal to Analog Pin # 0 on the Arduino. The value of this signal changes depending on the setting of the dial on the potentiometer. This analog reading is then used in the code you load onto the Arduino and effects the power signal in the LED circuit. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pin # 13: Power source, PWM (if code uses analogWrite) or digital (if code uses digitalWrite) output from Arduino board.

Arduino Analog Pin # 0: Analog input to Arduino board.

330 Ohm Resistor: A resistor resists the current flowing through the circuit. In the LED circuit it reduces the current so the LED in the circuit does not burn out.

LED: As in other diodes, current flows easily from the + side, or anode (longer wire), to the - side, or cathode (shorter wire), but not in the reverse direction.

Potentiometer: A voltage divider which outputs an analog value.

+5V: Five Volt power source.

Gnd: Ground

Code:

```
int sensorPin = 0;
int ledPin = 13;
int sensorValue = 0;
```

```
void setup() {
  pinMode(ledPin, OUTPUT);
}
```

```
void loop() {
```

```
//this line assigns whatever the analog Pin 0 reads to
sensorValue
```

```
sensorValue = analogRead(sensorPin);
```

```
digitalWrite(ledPin, HIGH);
delay(sensorValue);
digitalWrite(ledPin, LOW);
delay(sensorValue);
}
```

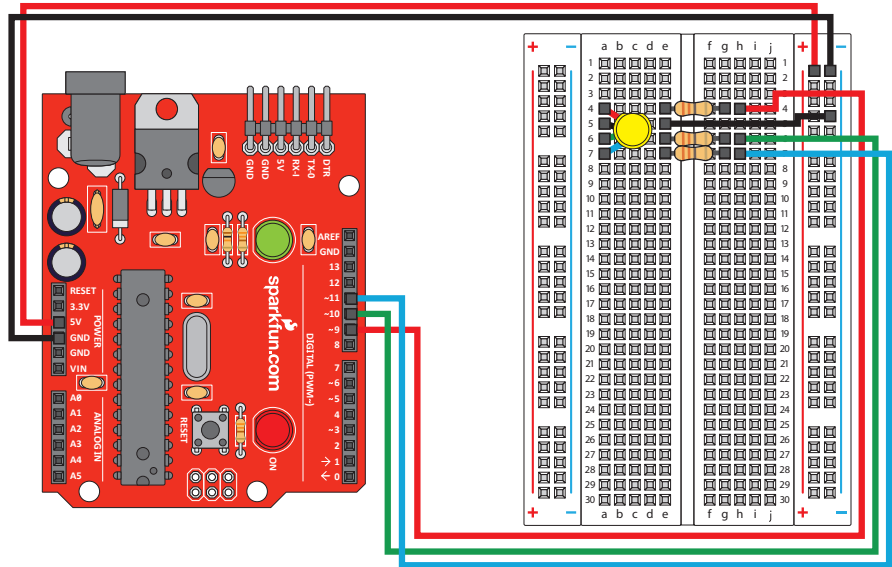
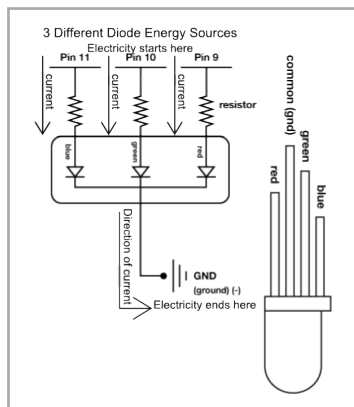
Additional thoughts: This is another example of input, only this time it is Analog. Circuits 2 and 5 in the S.I.K. introduces you to the two kinds of input your board can receive: Digital and Analog. Not sure what a voltage divider is? Check out the Voltage Divider page towards the back of this section.

CHAPTER 2

How the Circuits Work

Name:
Date:

Circuit #3: RGB LEDs



Explanation:

This circuit is pretty straight forward. The Digital Arduino Pins # 9, # 10 and # 11 supply a PWM value to each of the three different LEDs within the Tri-Color LED (Red, Green, and Blue). The LEDs are connected to the circuit so electricity enters through the anode (+, or longer wire) and exits through the cathode (-, or shorter wire). The resistors dissipate current so the LEDs do not draw current above the maximum rating and burn out. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground. By supplying different values to just these three Digital Pins you can mix 16,777,216 different colors!

Components:

Arduino Digital Pin # 9, # 10 and # 11: Power source, PWM output from Arduino board.

RGB LED: Unlike single color LEDs, on RGB (Also called 'Tri-Color') LEDs, the cathode (or ground wire) is the longest wire and each color (Red, Green, and Blue) gets its own lead. (See the schematic for details).

330 Ohm Resistor: A resistor resists the current flowing through the circuit. In this circuit the resistor reduces the current so the LEDs do not burn out.

Gnd: Ground

Code:

```
const int RED_LED_PIN = 9;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 11;
int redIntensity = 0;
int greenIntensity = 0;
int blueIntensity = 0;
const int DISPLAY_TIME = 100;

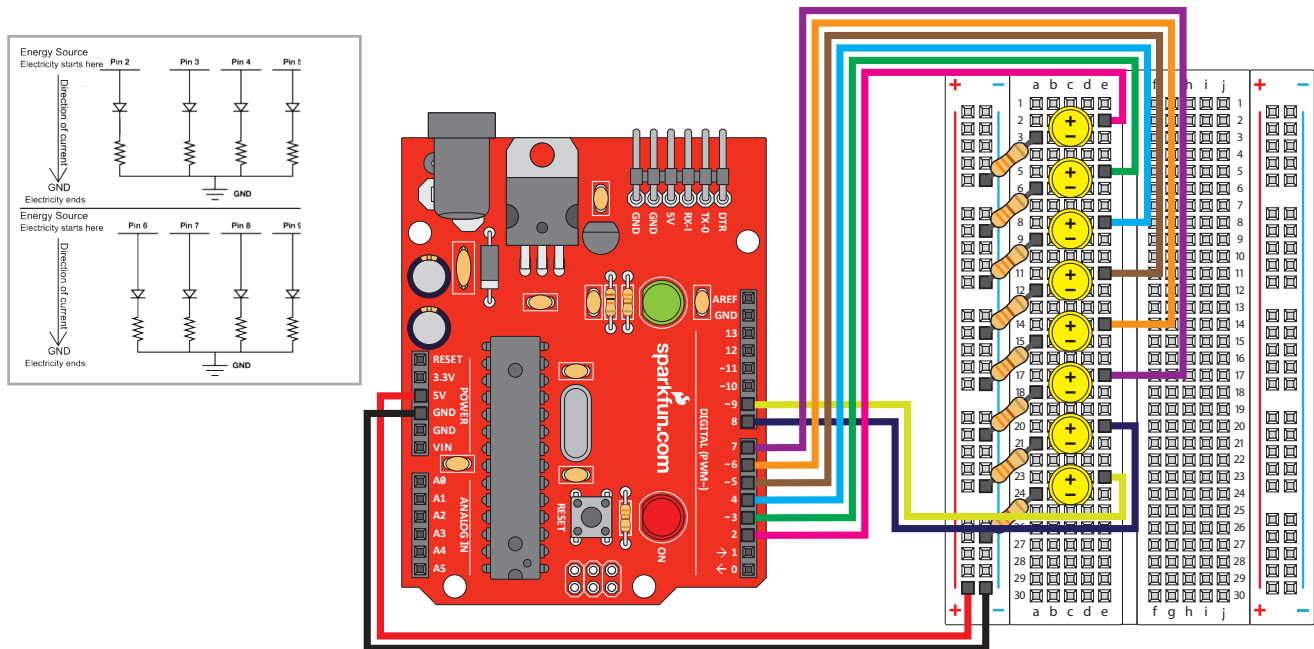
void setup() {
  // No setup required but you still need it
}

void loop(){
  for (greenIntensity = 0; greenIntensity <= 255;
  greenIntensity+=5) {
    redIntensity = 255-greenIntensity;
    analogWrite(GREEN_LED_PIN, greenIntensity);
```

```
    analogWrite(RED_LED_PIN, redIntensity);
    delay(DISPLAY_TIME);
  }
  for (blueIntensity = 0; blueIntensity <= 255;
  blueIntensity+=5) {
    greenIntensity = 255-blueIntensity;
    analogWrite(BLUE_LED_PIN, blueIntensity);
    analogWrite(GREEN_LED_PIN, greenIntensity);
    delay(DISPLAY_TIME);
  }
  for (redIntensity = 0; redIntensity <= 255; redIntensity+=5)
  {
    blueIntensity = 255-redIntensity;
    analogWrite(RED_LED_PIN, redIntensity);
    analogWrite(BLUE_LED_PIN, blueIntensity);
    delay(DISPLAY_TIME);
  }
}
```

Additional thoughts: See how combining just three simple outputs can create some amazing results?

Circuit #4: Multiple LEDs



Explanation:

This circuit takes electricity from Pin # 2 through Pin # 9 on the Arduino. The LEDs are connected to the circuit so electricity enters through the anode (+, or longer wire) and exits through the cathode (-, or shorter wire). The resistor dissipates current so the LEDs do not draw current above the maximum rating and burn out. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pins # 2 - # 9: Power source, analog (if code uses analogWrite, only possible on pins 3, 5, 6, & 9) or digital (if code uses digitalWrite) output from Arduino board.

LEDs: As in other diodes, current flows easily from the + side, or anode (longer wire), to the - side, or cathode (shorter wire), but not in the reverse direction. Also lights up!

330 Ohm Resistor: The resistors resist the current flowing through the circuit. In this circuit the resistors reduce the current so the LEDs do not burn out.

Gnd: Ground

Code:

```
//this line below declares an array
int ledPins[ ] = {2,3,4,5,6,7,8,9};

void setup( ) {

//these two lines set Digital Pins # 0 – 8 to output
for(int i = 0; i < 8; i++){
pinMode(ledPins[i],OUTPUT);

}
```

```
void loop( ) {

//these lines turn the LEDs on and then off
for(int i = 0; i <= 7; i++){
digitalWrite(ledPins[i], HIGH);
delay(delayTime);
digitalWrite(ledPins[i], LOW);
}

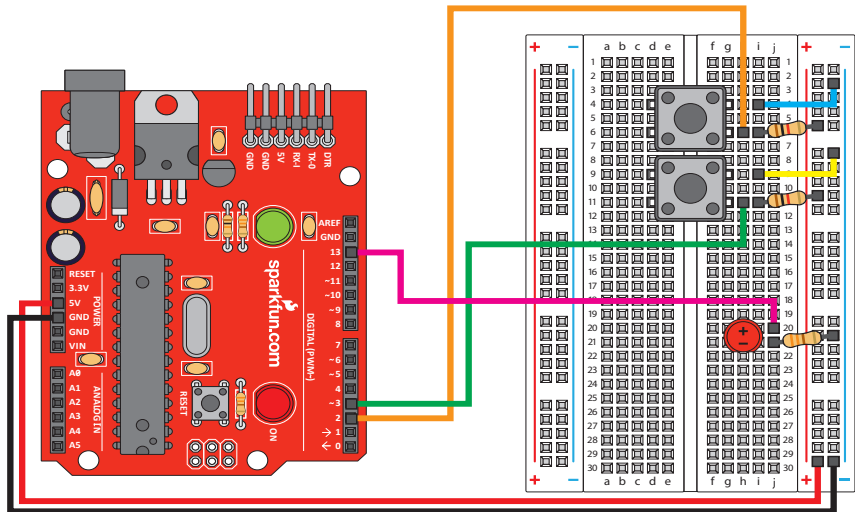
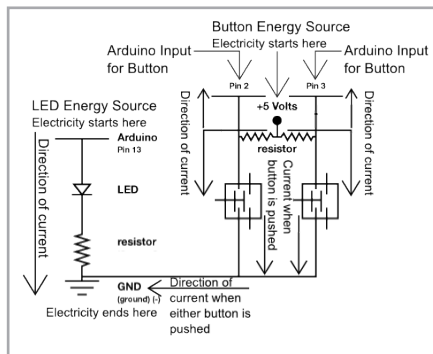
}
```

Additional thoughts: The code examples in the S.I.K get a little complicated for the fourth circuit, but don't worry, it's just more outputs. Some of the code examples use "for" loops to do something a number of times, if you're not familiar with "for" see Loops in Programming Concepts.

CHAPTER 2

How the Circuits Work

Circuit #5: Push Buttons



Explanation:

This circuit is actually two different circuits. One circuit for the buttons and another for the LED. See 'How the Circuits Work' Circuit 1 for an explanation of the LED circuit. The button circuit gets electricity from the 5V on the Arduino. The electricity passes through a pull up resistor, causing the input on Arduino Pins # 2 and # 3 to read HIGH when the buttons are not being pushed. When a button is pushed it allows the current to flow to ground, causing a LOW reading on the input pin connected to it. This LOW reading is then used in the code you load onto the Arduino and effects the power signal in the LED circuit.

Components:

Arduino Digital Pin # 13: Power source, PWM (if code uses analogWrite) or digital (if code uses digitalWrite) output from Arduino board.

Code:

```
const int buttonPin = 2;
const int ledPin = 13;

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  //this line below declares the button pin as input
  pinMode(buttonPin, INPUT);
}
```

Arduino Digital Pin # 2 and # 3: Digital input to Arduino board.

330 & 10K Ohm Resistors: Resistors resist the current flowing through the circuit. In the LED circuit the 330 ohm resistor reduces the current so the LED in the circuit does not burn out. In the button circuits the 10Ks ensure that the buttons will read HIGH when they are not pressed.

LED: As in other diodes, current flows easily from the + side, or anode (longer wire), to the - side, or cathode (shorter wire), but not in the reverse direction. Lights up!

Button: A press button which is open (or disconnected) when not in use and closed (or connected) when pressed. This allows you to complete a circuit when you press a button.

+5V: Five volt power source.

Gnd: Ground

```
void loop(){
  //this line assigns whatever the Digital Pin 2 reads to
  //buttonState
  buttonState = digitalRead(buttonPin);

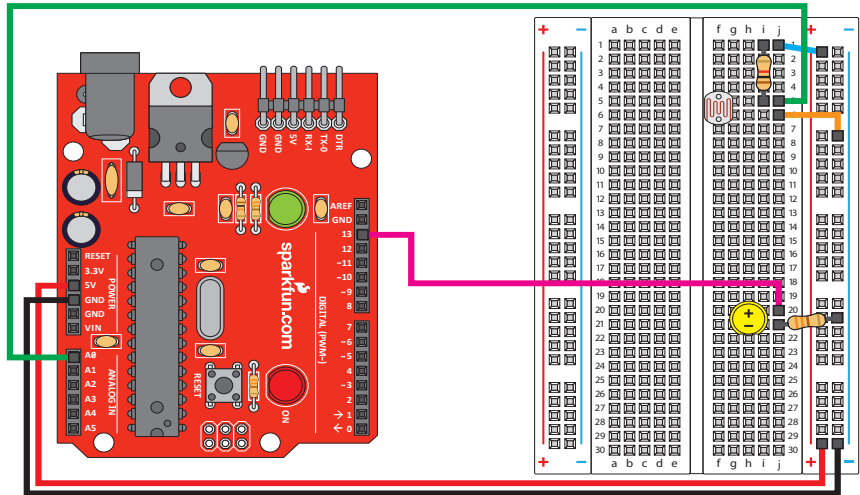
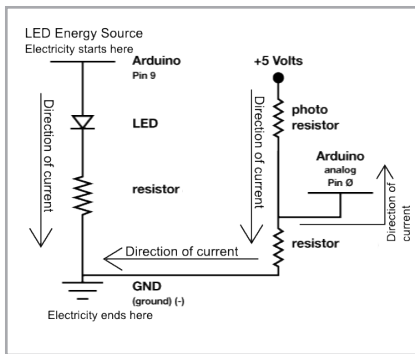
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

Additional thoughts: This circuit is the first circuit to use the input capabilities of the Arduino. Notice the difference in setup(). You are still using a Digital Pin but you are using it as input rather than output. Buttons are sweet by the way, let the kids press these buttons instead of yours.

CHAPTER 2

How the Circuits Work

Circuit #6: Photo Resistor



Explanation:

This circuit is actually two different circuits. One circuit for the photoresistor and another for the LED. See 'How the Circuits Work' Circuit 1 for an explanation of the LED circuit. The photoresistor circuit gets electricity from the 5V on the Arduino. The electricity passes through the photoresistor and sends a signal to Analog Pin # 0 on the Arduino. The value of this signal changes depending on the amount of sunlight. This analog reading is then used in the code you load onto the Arduino and effects the power signal in the LED circuit. The resistor below the Analog Pin connection creates the voltage divider necessary to measure the resistance of the photoresistor. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pin # 13: Power source, PWM (if code uses analogWrite) or digital (if code uses digitalWrite) output from Arduino board.

Arduino Analog Pin # 0: Analog input to Arduino board.

330 Ohm Resistor: A resistor resists the current flowing through the circuit. In the LED circuit it reduces the current so the LED in the circuit does not burn out. In the photoresistor circuit the resistor completes the voltage divider.

LED: As in other diodes, current flows easily from the + side, or anode (longer wire), to the - side, or cathode (shorter wire), but not in the reverse direction. Lights up!

Photoresistor: A resistor with a resistance value that changes depending on the amount of light hitting the sensor.

+5V: Five Volt power source.

Gnd: Ground

Code:

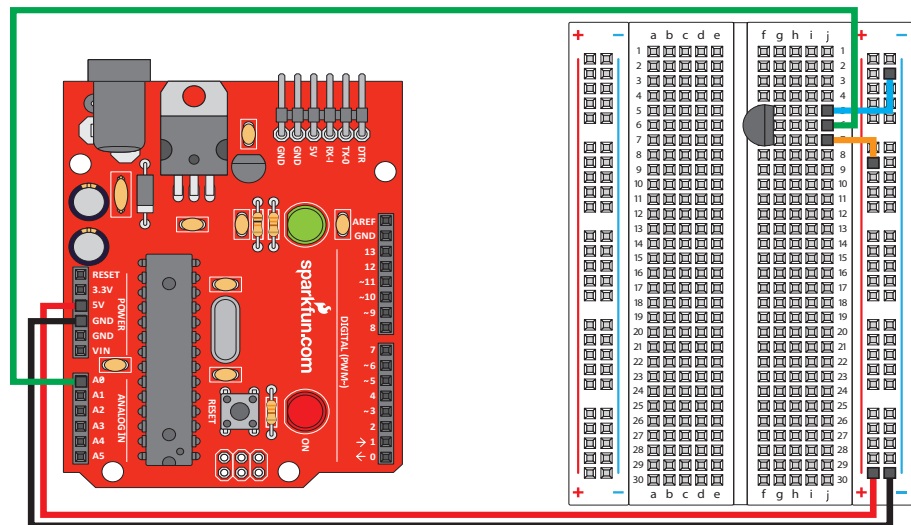
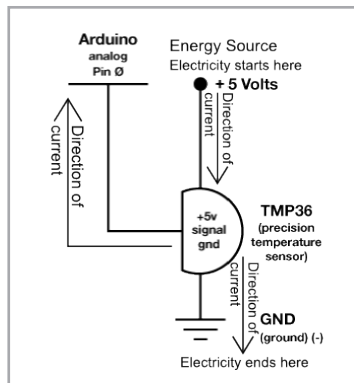
```
int lightPin = 0;
int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
}
```

```
void loop() {
  int lightLevel = analogRead(lightPin);
  lightLevel = map(lightLevel, 0, 900, 0, 255);
  lightLevel = constrain(lightLevel, 0, 255);
  analogWrite(ledPin, lightLevel);
}
```

Additional thoughts: This circuit is another example of Analog input. It is also a perfect example of a voltage divider. Don't worry about the "map" and "constrain" functions they are explained in the glossary. Unsure about the voltage divider? See the voltage divider page towards the back of this section.

Circuit #7: Temperature Sensor



Explanation:

This circuit takes electricity from the 5V on the Arduino. The temperature sensor sends an analog value to Arduino Analog Pin # 0. Then the electricity reaches ground, closing the circuit and allowing electricity to flow from power source through the sensor to ground. Finally Arduino uses its Serial monitor to display the temperature reading.

Components:

Arduino Analog Pin # 0: Analog input to Arduino board.

Temperature Sensor: Provides a voltage value depending on the temperature. Some math is then required to convert this value to Celsius or Fahrenheit.

+5V: Five Volt power source.

Gnd: Ground

Code:

```
int temperaturePin = 0;
```

```
void setup() {
```

```
//Serial comm. at a Baud Rate of 9600
  Serial.begin(9600);
}
```

```
void loop() {
```

```
//Calls the function to read the sensor pin
  float temp = getVoltage(temperaturePin);
```

```
//Below is a line that compensates for an offset
//(see datasheet)
  temp = (temp - .5) * 100;
```

```
//This line displays the variable temperature after all
//the math
  Serial.println(temp);
  delay(1000);
}
```

```
//function that reads the Arduino pin and starts to convert
//it to degrees
```

```
float getVoltage(int pin) {
  return (analogRead(pin) * .004882814);
}
```

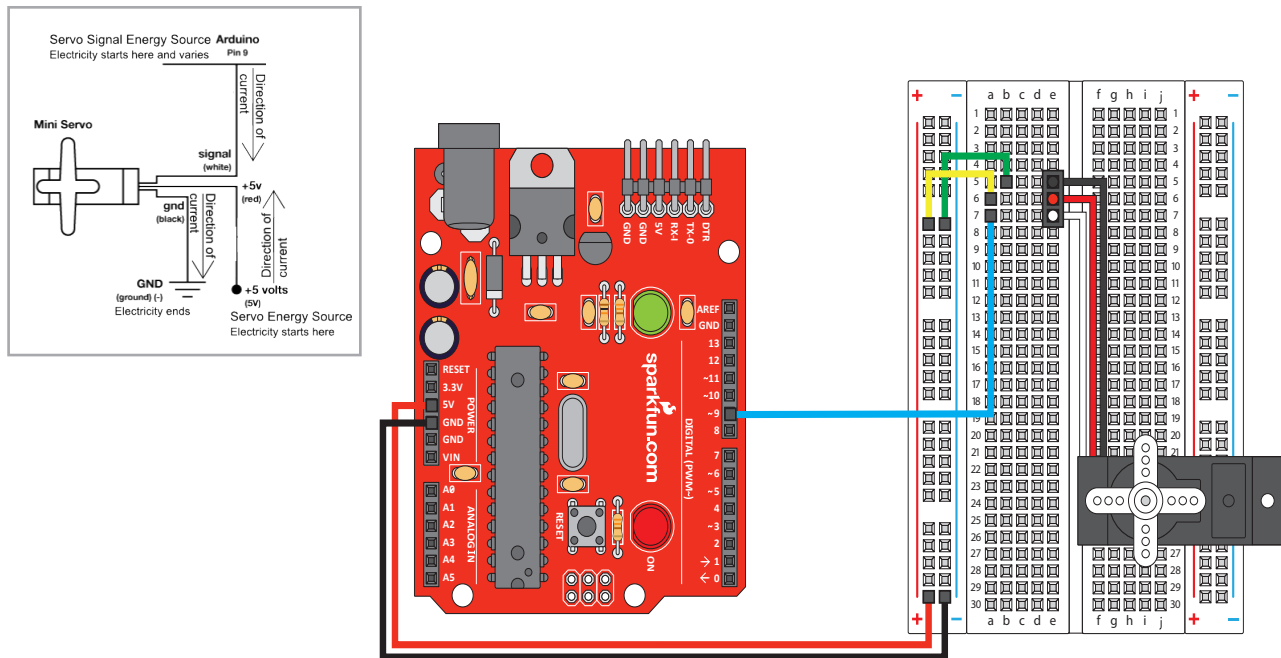
Additional thoughts: There is a lot of math involved in the code section of this circuit and it all has a reason. But how would you know you need to offset the temperature reading by .5 unless you had read the Datasheet? Also, pay attention to the code lines that enable Serial communication.

CHAPTER 2

How the Circuits Work

Name: _____
Date: _____

Circuit #8: A Single Servo



Explanation:

The servo in this circuit takes electricity from 5V on the Arduino. Pin # 9 on the Arduino supplies a PWM signal which sets the position of the servo. Each voltage value has a distinct correlating position. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pin #9: Signal power source for servo.

Servo: Sets the position of the servo arm depending on the voltage of the signal received.

+5V: Five volt power source.

Gnd: Ground

Code:

```
//include the servo library for use
#include <Servo.h>
Servo myservo; //create servo object

int pos = 0;

void setup() {
  myservo.attach(9);
}

void loop() {
```

```
//moves servo from 0° to 180°
for(pos = 0; pos < 180; pos += 1) {
  myservo.write(pos);
  delay(15);
}

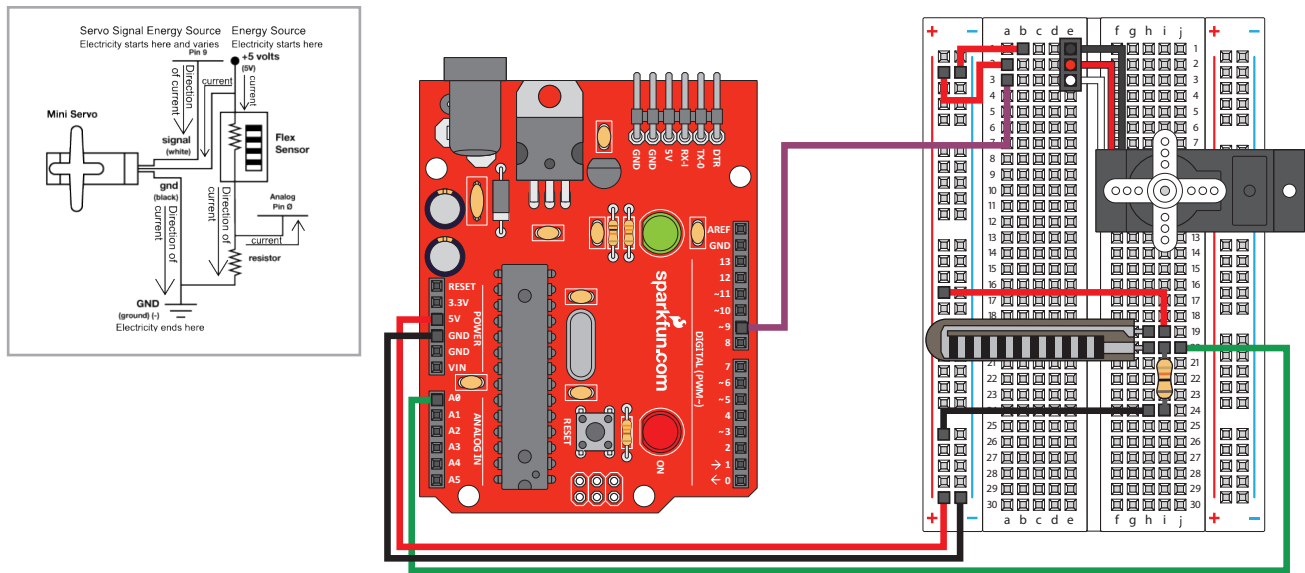
// moves servo from 180° to 0°
for(pos = 180; pos >= 1; pos -= 1) {
  myservo.write(pos);
  delay(15);
}
```

Additional thoughts: To some kids this is exciting stuff. There are all kinds of things kids can think to do with servos, you've just got to ask them. Throw out the word "robot" and see what comes back at you. Remember, this is just slightly more complicated output, same as the motor and LED.

CHAPTER 2

How the Circuits Work

Circuit #9: Flex Sensor



Explanation:

This circuit is actually two different circuits. One circuit for the flex sensor and another for the servo. See 'How the Circuits Work' Circuit 8 for an explanation of the servo circuit. The flex sensor circuit gets electricity from the 5V on the Arduino. The electricity passes through the flex sensor and sends a signal to Analog Pin # 0 on the Arduino. The value of this signal changes depending on the amount of bend in the flex sensor. This analog reading is then used in the code you load onto the Arduino and sets the position of the servo. The resistor and flex sensor create a voltage divider which is measured by Analog Pin # 0. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pin # 9: Power source, PWM output from Arduino board.

Arduino Analog Pin # 0: Analog input to Arduino board.

10K Ohm Resistor: A resistor resists the current flowing through the circuit.

Flex Sensor: A resistor with a value that varies depending on the amount of bend in the sensor.

Servo: Sets the position of the servo arm depending on the voltage of the signal received.

+5V: Five Volt power source.

Gnd: Ground

Code:

```
#include <Servo.h> //include the servo library
Servo myservo;

int potpin = 0; //sets pin 0 to read the flex sensor
int val;

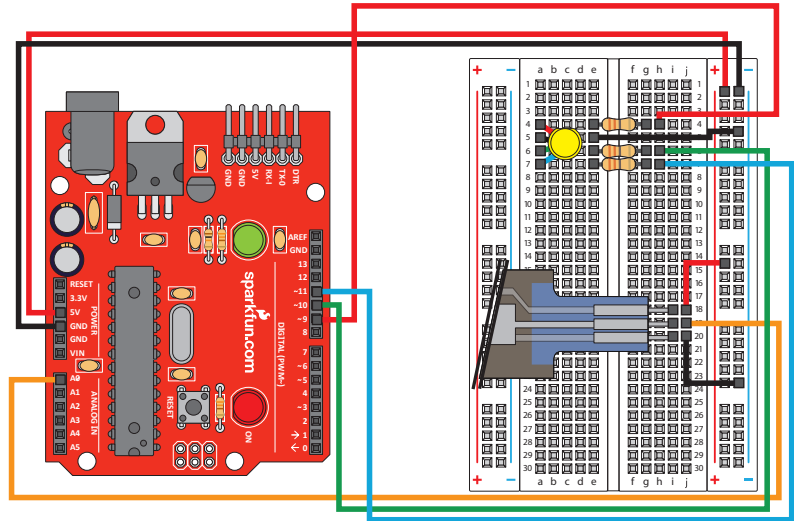
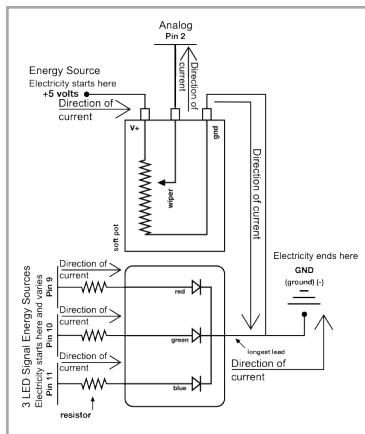
void setup() {
  Serial.begin(9600);
```

```
  myservo.attach(9);
}

void loop() {
  val = analogRead(potpin); //get a reading from the flex sensor
  Serial.println(val);
  val = map(val, 50, 300, 0, 179);
  myservo.write(val);
  delay(15);
}
```

Additional thoughts: This analog input is definitely very different from any other input we have looked at so far but the concept is the same. We treat the sensor as a resistor in a voltage divider to get a reading and then change our output depending on that reading.

Circuit #10: Soft Potentiometer



Explanation:

This circuit is actually two different circuits. One circuit for the soft pot and another for the RGB LED. See 'How the Circuits Work' Circuit 3 for an explanation of the RGB LED circuit. The soft pot circuit gets electricity from the 5V on the Arduino. The electricity passes through the soft pot and sends a signal out the com line of the soft pot to Analog Pin # 0 on the Arduino. The value of this signal changes depending on where the wiper (any type of contact) touches the soft pot. This analog reading is then used in the code you load onto the Arduino and sets the color of the RGB LED. Notice that yet again our sensor and the input pin form a voltage divider, only this time the voltage divider is completely inside the sensor. The wiper divides the resistor into two different portions with values that depend on the position of the wiper. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pins # 9, 10, 11: Power source, PWM output from Arduino board.

Arduino Analog Pin # 0: Analog input to Arduino board.

330 Ohm Resistor: A resistor resists the current flowing through the circuit. In the RGB LED circuit it reduces the current so the LED it is attached to does not burn out.

Flex Sensor: A resistor with a value that varies depending on the amount of bend in the sensor.

RGB LED: A grouping of three LEDs, Red, Green and Blue. Power goes in three different anodes (+, the short wires) and out one common cathode (-, the long wire). Lights up!

+5V: Five Volt power source.

Gnd: Ground

Code:

```
const int RED_LED_PIN = 9;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 11;

void setup() {
  //No setup necessary but you still need it
}

void loop() {
  int sensorValue = analogRead(0);
```

```
  int redValue = constrain(map(sensorValue, 0, 512, 255, 0),0,255);
  int greenValue = constrain(map(sensorValue, 0, 512, 0, 255),0,255)-constrain(map(sensorValue, 512, 1023, 0, 255),0,255);
  int blueValue = constrain(map(sensorValue, 512, 1023, 0, 255),0,255);

  analogWrite(RED_LED_PIN, redValue);
  analogWrite(GREEN_LED_PIN, greenValue);
  analogWrite(BLUE_LED_PIN, blueValue);
}
```

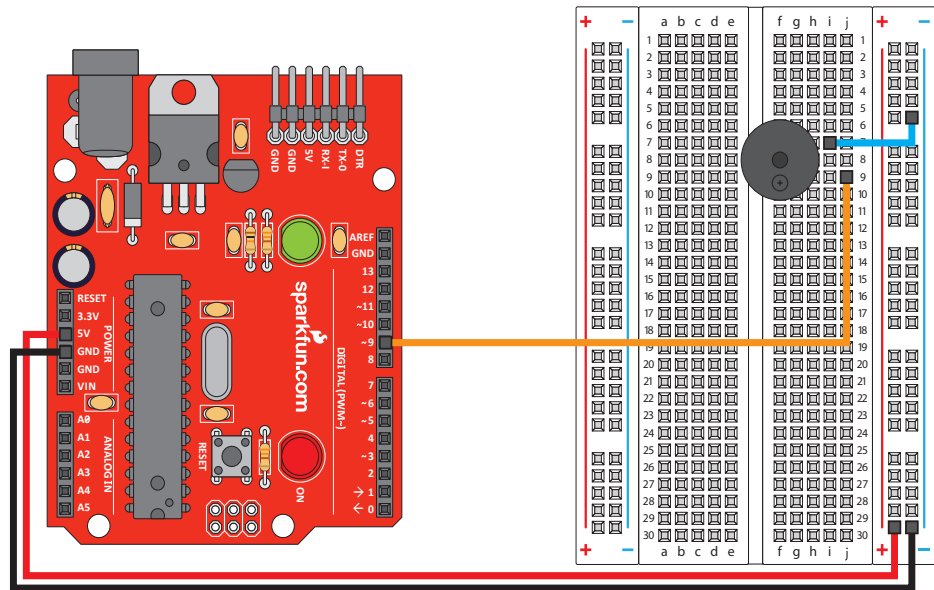
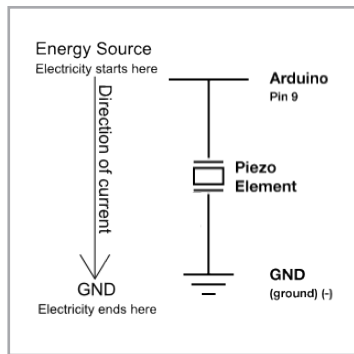
Additional thoughts: This analog sensor is similar to the flex sensor. You will often see the basic concepts covered in the S.I.K. in different forms as you work with more complicated sensors and outputs. Most of these technologies are built using the same building blocks and some math.

CHAPTER 2

How the Circuits Work

Circuit #11: Piezo Elements

Name: _____
Date: _____



Explanation:

This circuit gets electricity from Arduino Pin # 9. The Piezo element plays different musical notes depending on the speed and duration of the electrical signal sent from Pin # 9. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pin # 9: Power source, digital output from Arduino board. (If changed to PWM output this creates distortion of note, not a change in volume.)

Piezo element: A tiny speaker with a magnetic coil that responds to electrical current by moving more or less depending on the current. The coil is attached to a diaphragm that moves air and causes the noise we hear.

Gnd: Ground

Note:

This section contains only the two functions needed to make the piezo play a note of a given duration. These functions are called in the loop () function.

Code:

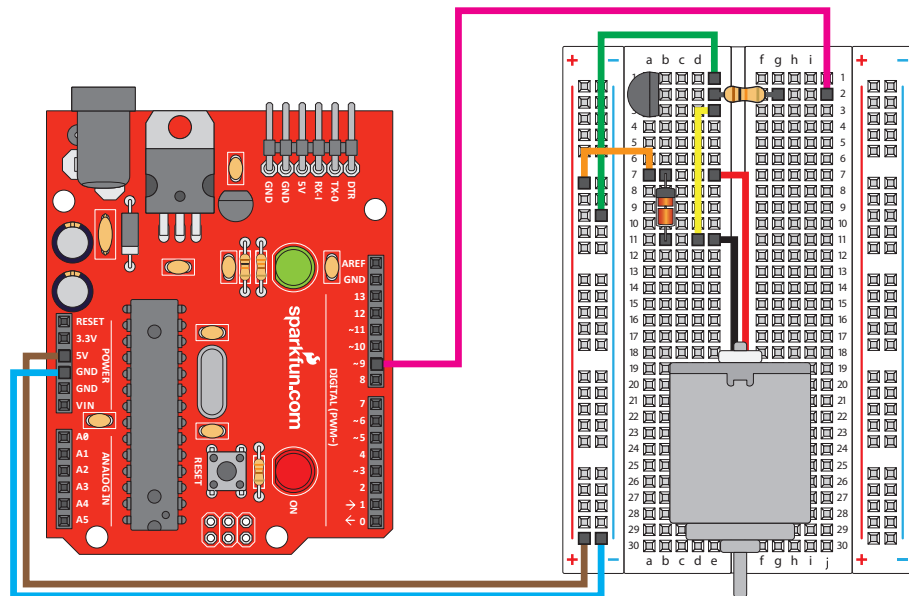
```
void playTone(int tone, int duration) {  
  for (long i = 0; i < duration * 1000L; i += tone * 2) {  
    digitalWrite(speakerPin, HIGH);  
    delayMicroseconds(tone);  
    digitalWrite(speakerPin, LOW);
```

```
    delayMicroseconds(tone);  
  }  
}  
void playNote(char note, int duration) {  
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };  
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136,  
    1014, 956 };  
  
  for (int i = 0; i < 8; i++) {  
    if (names[i] == note) {  
      playTone(tones[i], duration);  
    }  
  }  
}
```

Additional thoughts: This code is fairly complicated. Don't worry if you don't understand some aspects of it. If you do understand it, congratulations! You are already ahead of this packet in regards to Arduino code. Just remember, this is another way to use digital pins to create analog output.

Circuit #12: Spinning a Motor

Date:



Components:

Gnd: Ground

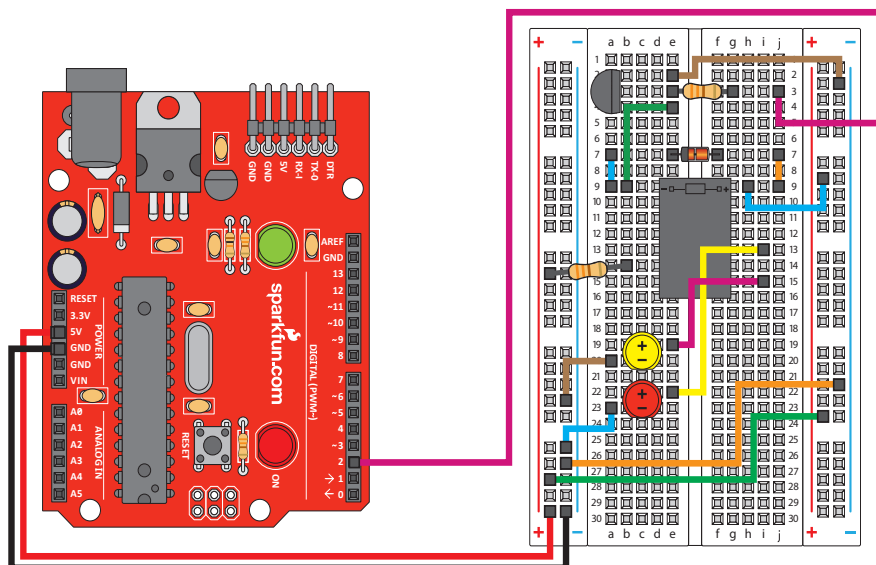
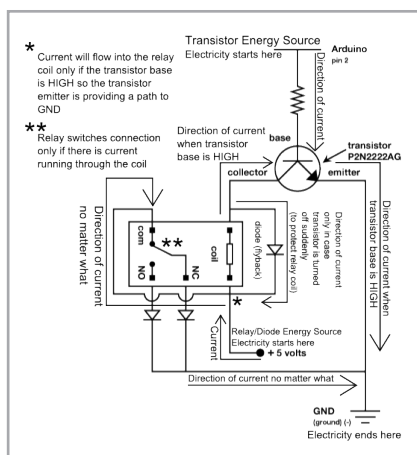
```
void loop() {
  for (int i = 0; i < 256; i++){
    analogWrite(motorPin, i);
    delay(50);
  }
}
```

SIK BINDER //36

CHAPTER 2

How the Circuits Work

Circuit #13: Relays



Explanation:

The relay circuit gets electricity from the 5V on the Arduino. The electricity always passes through the relay communication line which is switched to either NO (Normally Open) or NC (Normally Closed), lighting up one of the two LEDs. The transistor gets electricity from Arduino Digital Pin # 2 with a resistor to prevent burn out. In this case the transistor receives a digital signal. The transistor closes the circuit when it is sent a HIGH value, allowing electricity to flow through the relay coil, into the collector, out the emitter and to ground, completing the circuit. The energized coil sets the relay switch to NO. The transistor opens or breaks the circuit when it is sent a LOW value, so no electricity passes through the coil and the relay switch is set to NC. The flyback diode connected close to the motor is simply to protect the motor in the rare case that electricity flows from the transistor towards the motor. This only happens if the transistor is shut off suddenly.

Components:

Arduino Digital Pin # 2: Power source, digital output from Arduino board.

Relay: The relay acts as an electrically operated switch between the two LED's.

Transistor: A semiconductor which can be used as an amplifier or a switch. In this case the amount of electricity supplied to the base corresponds to the amount of electricity allowed through from the collector to the emitter.

330 Ohm & 10K Resistors: A resistor resists the current flowing through the circuit. In the transistor circuit it reduces the current so the transistor in the circuit does not burn out.

Flyback Diode: As in other diodes, current flows easily from the + side, or anode, to the - side, or cathode, but not in the reverse direction. In this case the diode is being used to prevent current from 'flying back' to the relay in case the transistor is suddenly turned off.

Code:

```
int ledPin = 2;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
```

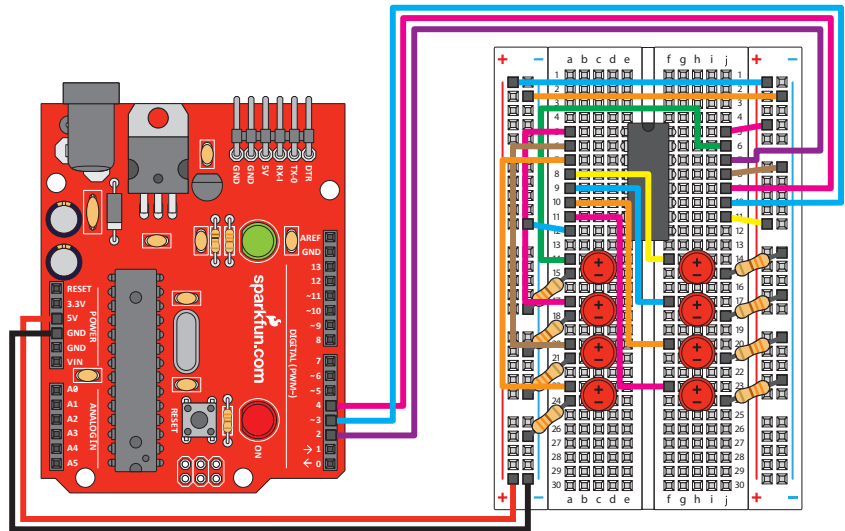
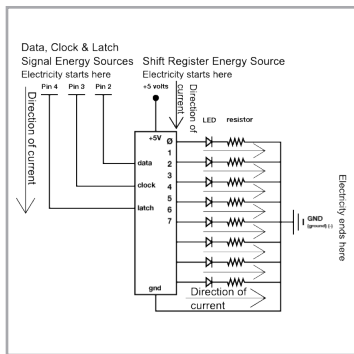
```
//set the transistor on
digitalWrite(ledPin, HIGH);
// wait for a second
delay(1000);
// set the transistor off
digitalWrite(ledPin, LOW);
// wait for a second
delay(1000);
}
```

Additional thoughts: By now you should be thinking that transistors seem pretty important in the world of electrical circuits. They can be used as switches or amplifiers and they are often called the most important invention of the 20th century. The relay is also a great control component, but it needs something to activate it, hence the transistor.

CHAPTER 2

How the Circuits Work

Circuit #14: Shift Register



Explanation:

The shift register in this circuit takes electricity from 5V on the Arduino. Pin # 2, # 3 and # 4 on the Arduino supply a digital value. The latch and clock pins are used to allow data into the shift register. The shift register sets the eight output pins to either HIGH or LOW depending on the values sent to it via the data pin. The LEDs are connected to the circuit so electricity enters through the anode (+, or longer wire) and exits through the cathode (-, or shorter wire) if the shift register pin is HIGH. The resistor dissipates current so the LEDs do not draw current above the maximum rating and burn out. Finally the electricity reaches ground, closing the circuit and allowing electricity to flow from power source to ground.

Components:

Arduino Digital Pin # 2, # 3 and # 4: Signal power source for data, clock and latch pins on shift register.

Shift register: Allows usage of eight output pins with three input pins, a power and a ground.

Link: <http://www.sparkfun.com/datasheets/Components/General/sn74hc165.pdf>

LED: As in other diodes, current flows easily from the + side, or anode (longer wire), to the - side, or cathode (shorter wire), but not in the reverse direction. Lights up!

330 Ohm Resistor: A resistor resists the current flowing through the circuit. In this circuit the resistor reduces the current so the LED does not burn out.

+5V: Five volt power source.

Gnd: Ground

Code:

```
int data = 2;
int clock = 3;
int latch = 4;
```

```
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;
```

```
void setup() {
  pinMode(data, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(latch, OUTPUT);
}
```

```
}

void loop(){
  for(int i = 0; i < 256; i++) {
    updateLEDs(i);
    delay(25);
  }
}

void updateLEDs(int value) {
  digitalWrite(latch, LOW);
  shiftOut(data, clock, MSBFIRST, value);
  digitalWrite(latch, HIGH);
}
```

Additional thoughts: For more advanced components you will need to read documentation or datasheets to figure out how to use them. Any documentation is good as long as you can get the correct information out of it. Datasheets are your friends!