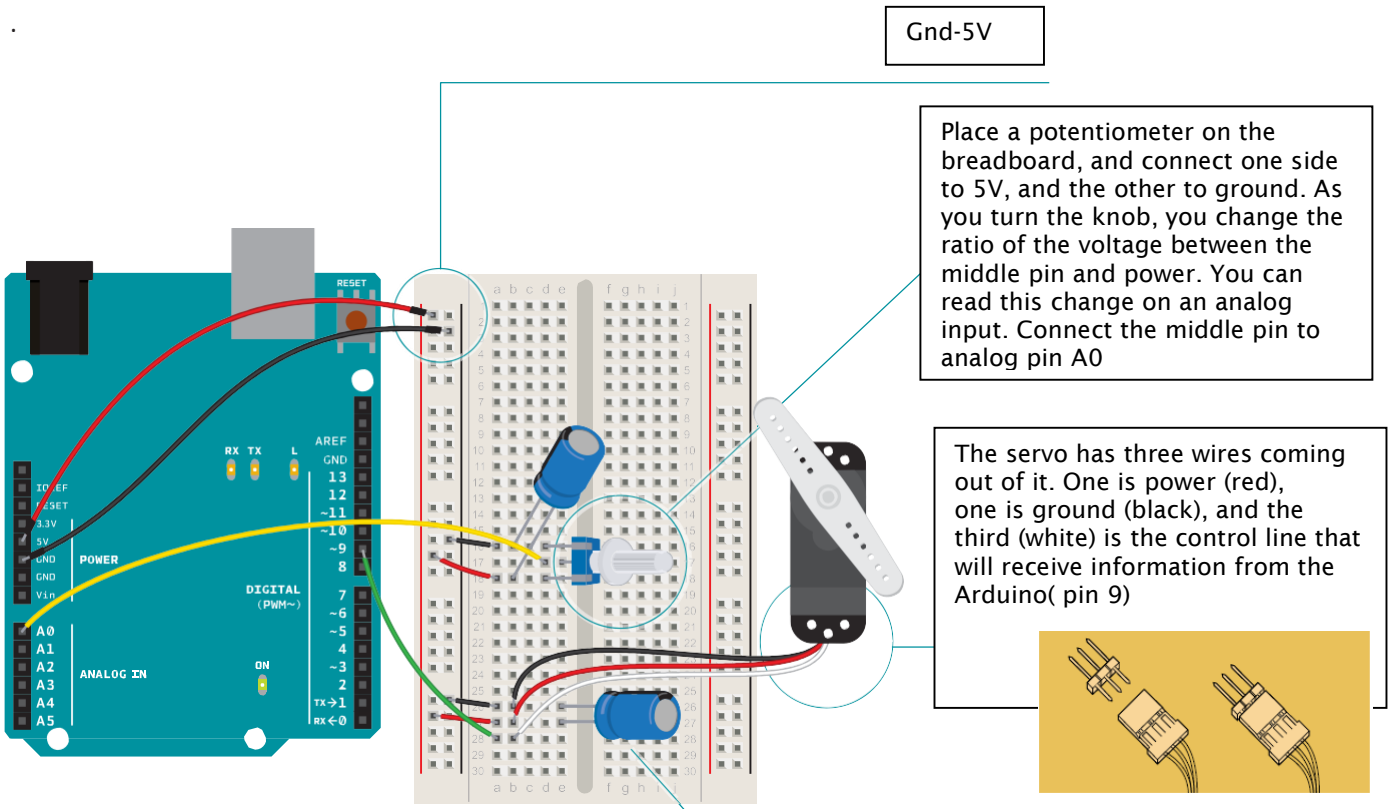
# Servo motor

Servo motors are a special type of motor that don't spin around in a circle, but move to a specific position and stay there until you tell them to move again.

Servos usually only rotate 180 degrees (one half of a circle). Similar to the way you used pulses to PWM an LED in the Color Mixing Lamp Project, servo motors expect a number of pulses that tell them what angle to move to. The pulses always come at the same time intervals, but the width varies between 1000 and 2000 microseconds. While it's possible to write code to generate these pulses, the Arduino software comes with a library that allows you to easily control the motor. Because the servo only rotates 180 degrees, and your analog input goes from 0-1023, you'll need to use a function called **map()** to change the scale of the values coming from the potentiometer

.



```
// include the Servo library
#include <Servo.h>

Servo myServo;  // create a servo object

int const potPin = A0; // analog pin used to connect the potentiometer
int potVal;  // variable to read the value from the analog pin
int angle;   // variable to hold the angle for the servo motor

void setup() {
  myServo.attach(9); // attaches the servo on pin 9 to the servo object
  Serial.begin(9600); // open a serial connection to your computer
}

void loop() {
  potVal = analogRead(potPin); // read the value of the potentiometer
  // print out the value to the Serial Monitor
  Serial.print("potVal: ");
  Serial.print(potVal);

  // scale the numbers from the pot
  angle = map(potVal, 0, 1023, 0, 179);

  // print out the angle for the servo motor
  Serial.print(", angle: ");
  Serial.println(angle);

  // set the servo position
  myServo.write(angle);

  // wait for the servo to get there
  delay(15);
}
```

To use the servo library, you'll first need to import it. This makes the additions from the library available to your sketch.

To refer to the servo, you're going to need to create a named instance of the servo library in a variable. This is called an object.

When you do this, you're making a unique name that will have all the functions and capabilities that the servo library offers. From this point on in the program, every time you refer to **myServo**, you'll be talking to the servo object.

Set up a named constant for the pin the potentiometer is attached to, and variables to hold the analog input value and angle you want the servo to move to.

In the setup(), you're going to need to tell the Arduino what pin your servo is attached to.

Include a serial connection so you can check the values from the potentiometer and see how they map to angles on the servo motor.

In the loop(), read the analog input and print out the value to the serial monitor.

To create a usable value for the servo motor from your analog input, it's easiest to use the **map()** function. This handy function scales numbers for you. In this case it will change values between 0-1023 to values between 0-179. It takes five arguments: the number to be scaled (here it's **potVal**), the minimum value of the input (0), the maximum value of the input (1023), the minimum value of the output (0), and the maximum value of the output (179). Store this new value in the angle variable.

Then, print out the mapped value to the serial monitor. Finally, it's time to move the servo. The command **myServo.write()** moves the motor to the angle you specify.
At the end of the loop() put a delay so the servo has time to move to its new position. Decide if it necessary to increase the delay value.