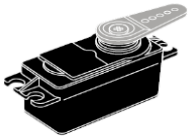
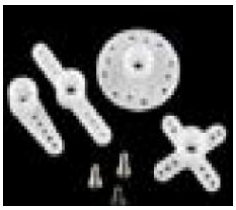
The “Simon Says” game uses LEDs to flash a pattern, which the player must remember and repeat using four buttons. This simple electronic game has been a classic since the late 1970s.

Grab the following quantities of each part listed to build this circuit:

NEW COMPONENTS

SERVO MOTORS: Regular DC motors have two wires. When you hook the wires up to power, the motor spins round and around. Servo motors, on the other hand, have three wires: one for power, one for ground and one for signal. When you send the right signal through the signal wire, the servo will move to a specific angle and stay there. Common servos rotate over a range of about 0° to 180°. The signal that is sent is a PWM signal, the same used to control the RGB LED in Project 1.

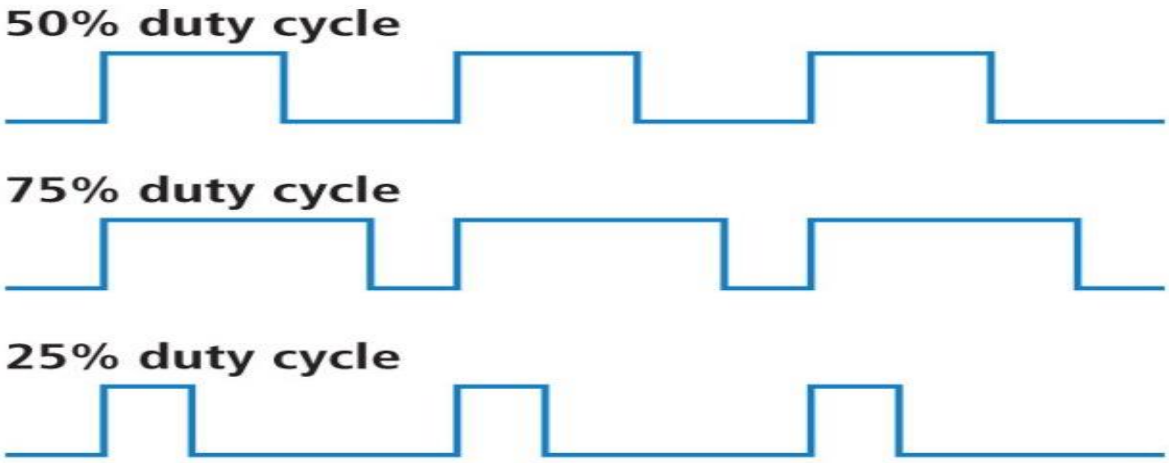


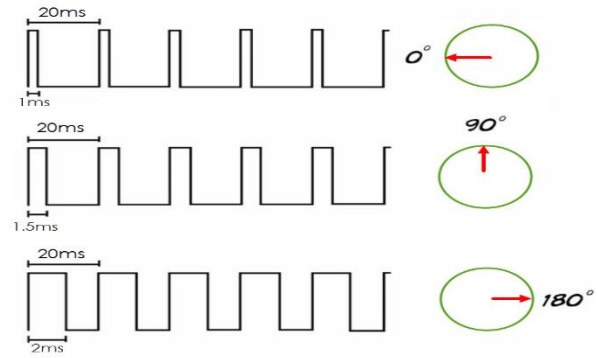


Included with your servo motor you will find a variety of motor mounts that connect to the shaft of your servo. You may choose to attach any mount you wish for this circuit. It will serve as a visual aid, making it easier to see the servo spin. The mounts will also be used at the end of this project.

NEW CONCEPTS

DUTY CYCLE: Pulse-Width Modulation (PWM) is a great way to generate servo control signals. The length of time a PWM signal is on is referred to as the duty cycle. Duty cycle is measured in percentage. Thus, a duty cycle of 50 percent means the signal is on 50 percent of the time. The variation in the duty cycle is what tells the servo which position to go to in its rotation.





ARDUINO LIBRARIES: Writing code that sends precise PWM signals to the servo would be time consuming and would require a lot more knowledge about the servo. Luckily, the Arduino IDE has hundreds of built-in and user-submitted containers of code called libraries. One of the built-in libraries, the Servo Library, allows us to control a servo with just a few lines of code! To use one of the built-in Arduino libraries, all you have to do is "include" a link to its header file. A header file is a smaller code file that contains definitions for all the functions used in that library. By adding a link to the header file in your code, you are enabling your code to use all of those library functions. To use the Servo Library, you would add the following line to the top of your sketch.

```
#include <Servo.h>
```

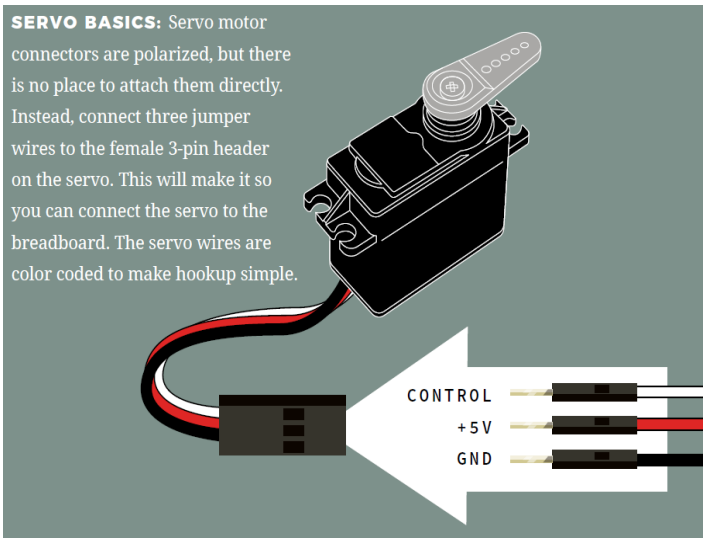
OBJECTS AND METHODS: To use the Servo Library, you will have to start by creating a servo object, like this:

```
Servo myServo;
```

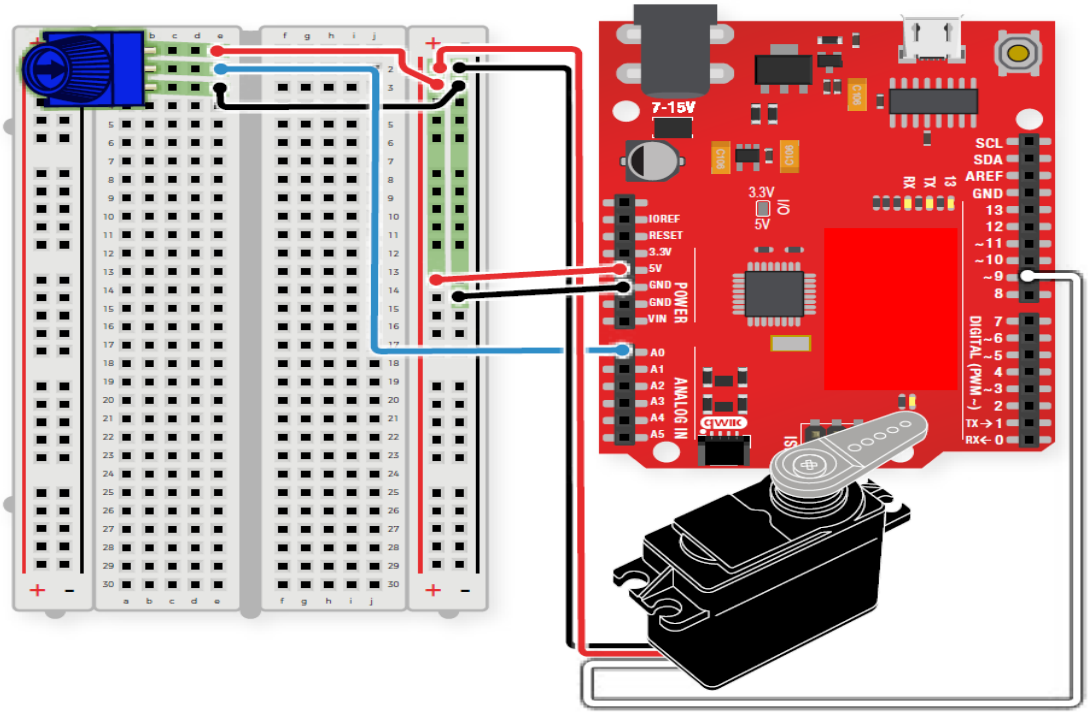
Objects look a lot like variables, but they can do much more. Objects can store values, and they can have their own functions, which are called methods. The most used method that a servo object has is .write():

```
myServo.write(90);
```

The write method takes one parameter, a number from 0 to 180, and moves the servo arm to the specified position (in this case, degree 90). Why would we want to go to the trouble of making an object and a method instead of just sending a servo control signal directly over a pin? First, the servo object does the work of translating our desired position into a signal the servo can read. Second,using objects makes it easy for us to add and control more than one servo.



HOOKUP GUIDE



JUMPER WIRES	A0 to E2 · 5V to 5V(+) · GND to GND(–) · E1 to 5V (+) · E3 to GND (-)
POTENTIOMETER	B1 + B2 + B3
SERVO LEADS	WHITE WIRE to D9 · RED WIRE to 5V(+) · BLACK WIRE to GND(–)

SOURCE CODE:

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

int potPosition;          //this variable will store the position of the potentiometer
int servoPosition;        //the servo will move to this position

Servo myservo;            //create a servo object

void setup() {

  myservo.attach(9);      //tell the servo object that its servo is plugged into pin 9

}

void loop() {

  potPosition = analogRead(A0);                  //use analog read to measure the position of the potentiometer (0-1023)

  servoPosition = map(potPosition, 0, 1023, 20, 160); //convert the potentiometer number to a servo position from 20-160
                                                 //Note: its best to avoid driving the little SIK servos all the
                                                 //way to 0 or 180 degrees it can cause the motor to jitter, which is bad for the servo.

  myservo.write(servoPosition);                  //move the servo to the 10 degree position
}
```