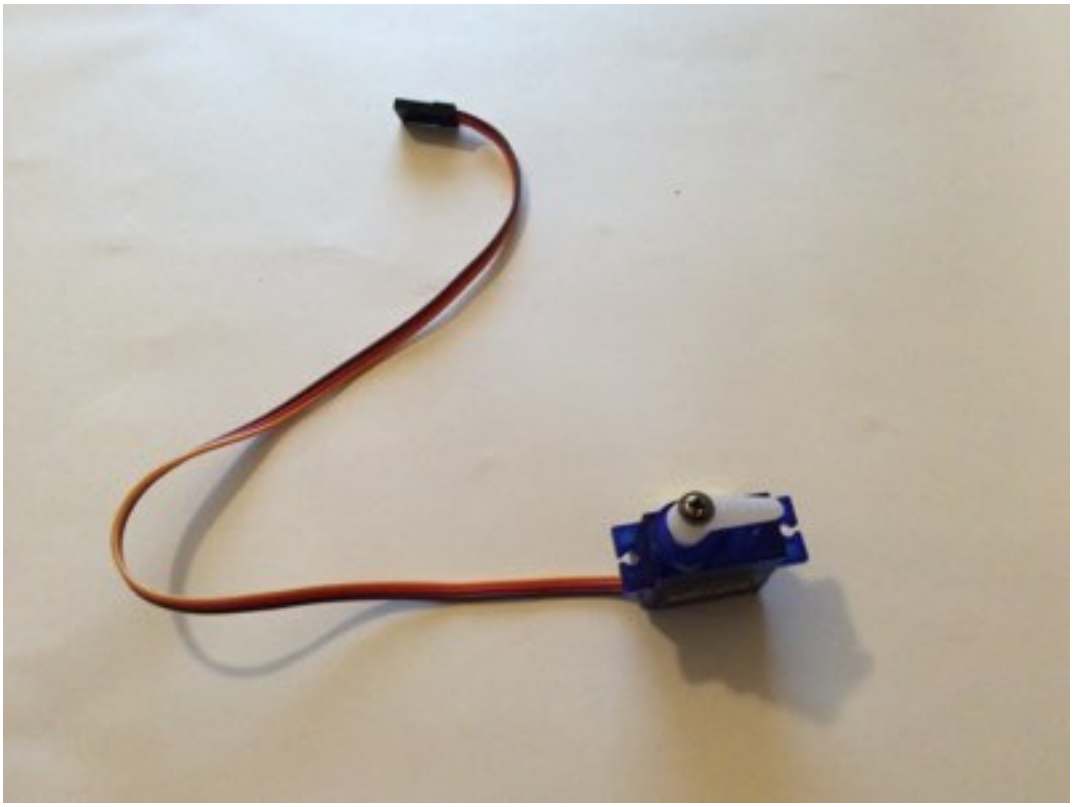
Working with Servo Motors

Servo motors are small RC hobby motors that can be controlled by an Arduino through controlling the shaft. Servos have plastic arms that connect to the top of this shaft. You'll come across servos that have different tops attached. The ones in your kit have some options for you to try out. You're able to switch out these tops rather easily- sometimes there is a tiny screw that needs to be detached, but often you can just use your fingers to switch out the arms.

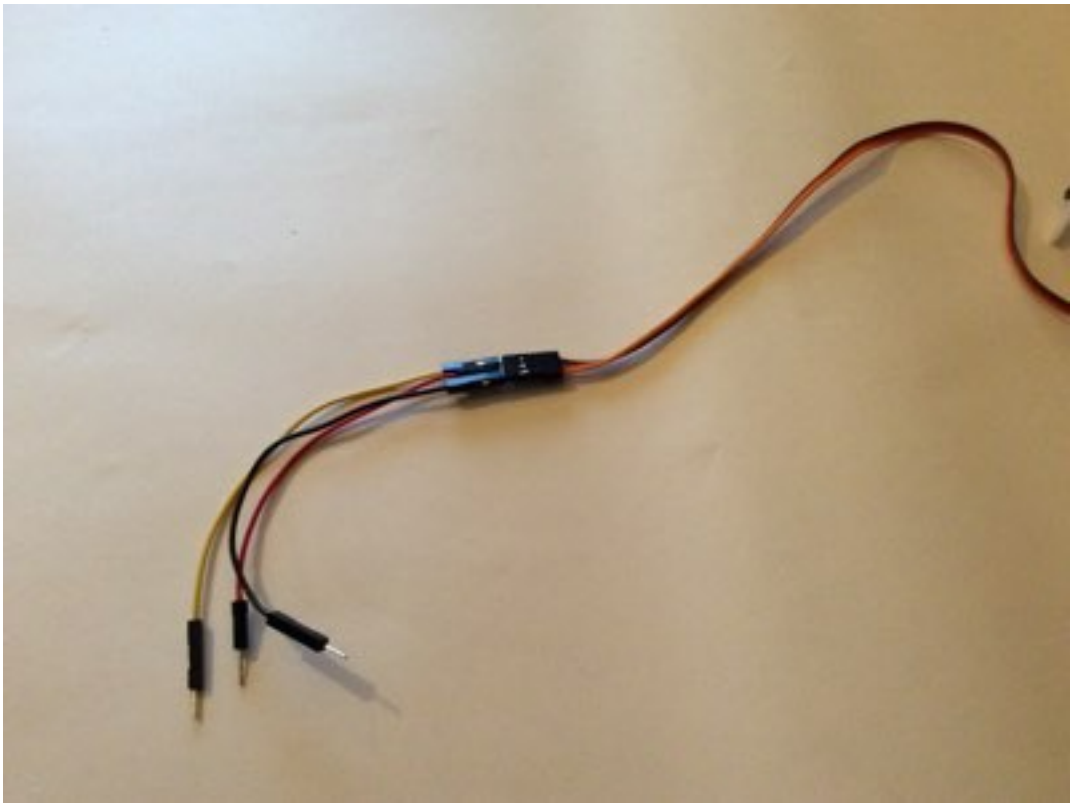


Controlling a Servo

Servo motors have three wires:

- Red - Power (5v)
- Brown - Ground
- Orange - Signal / Data

These wires then connect to the Arduino board. The Arduino sends electric pulse signals to the servo, which causes it to rotate. When you begin working with the code, you'll see how this process of sending a message from the Arduino to a component is quite similar to what you've done so far with LEDs.



Servos and Arduino

Once the servo motor is connected to an Arduino board, you're able to control the angle of the shaft position, typically between a range of *0 to 180 degrees*. There are certain types of servos that are *continuous rotation* that can move at different speeds, but the more common types work within a range of 0 to 180 degrees and move one step (degree) at a time.

In order to work with servos, you'll need to import a code *library* specifically for the servo. There are several Arduino libraries that you're able to include in projects. Often these libraries are written to provide additional support or functionality for working with certain components.

Practicing importing and including libraries will open up lots of projects and ideas!

The *Servo Library* can support up to **12** motors on the board, and uses a special **servo.write()** command to send position data to the motor. The **servo.write()** is from the Servo Library and the Arduino IDE wouldn't understand what it means without including the library.

You'll have the entirety of the code to experiment with, but let's cover some key parts:

Including a Library: to include a library in your code, you place an *include* at the top of your code, before anything else:

```
#include <Servo.h> //This is necessary when working
```

The name of the library goes in the < >. In this example, it's called *Servo.h*

Many times when working with libraries there are code examples that you can reference if you need help knowing what to include.

Creating a servo object: the next key piece of code creates a servo object. This is what the Arduino uses to work with the servo component:

```
Servo myservo;  // create servo object to control a servo
```

After you've done this step, you're then able to interact with the servo that you've attached to your board.

Connecting the servo: Just like when working with any of the other components, such as the LED, you need to tell the Arduino the pin

number where the component is attached. For the servos, this is the pin that the orange (signal) wire connects with. Here's the code that goes into the **void setup()**:

```
myservo.attach(9);  // attaches the servo on pin 9 to the se
```

That covers the key parts of the code. For the project, you'll have code provided that you can experiment with and tweak.

Activity: Setting up a Servo

For this activity, you're going to practice the basic servo connection and experiment with the code. Once you have the basic concept, you'll then be able to integrate servos into more of your projects as well as combine them with other components.

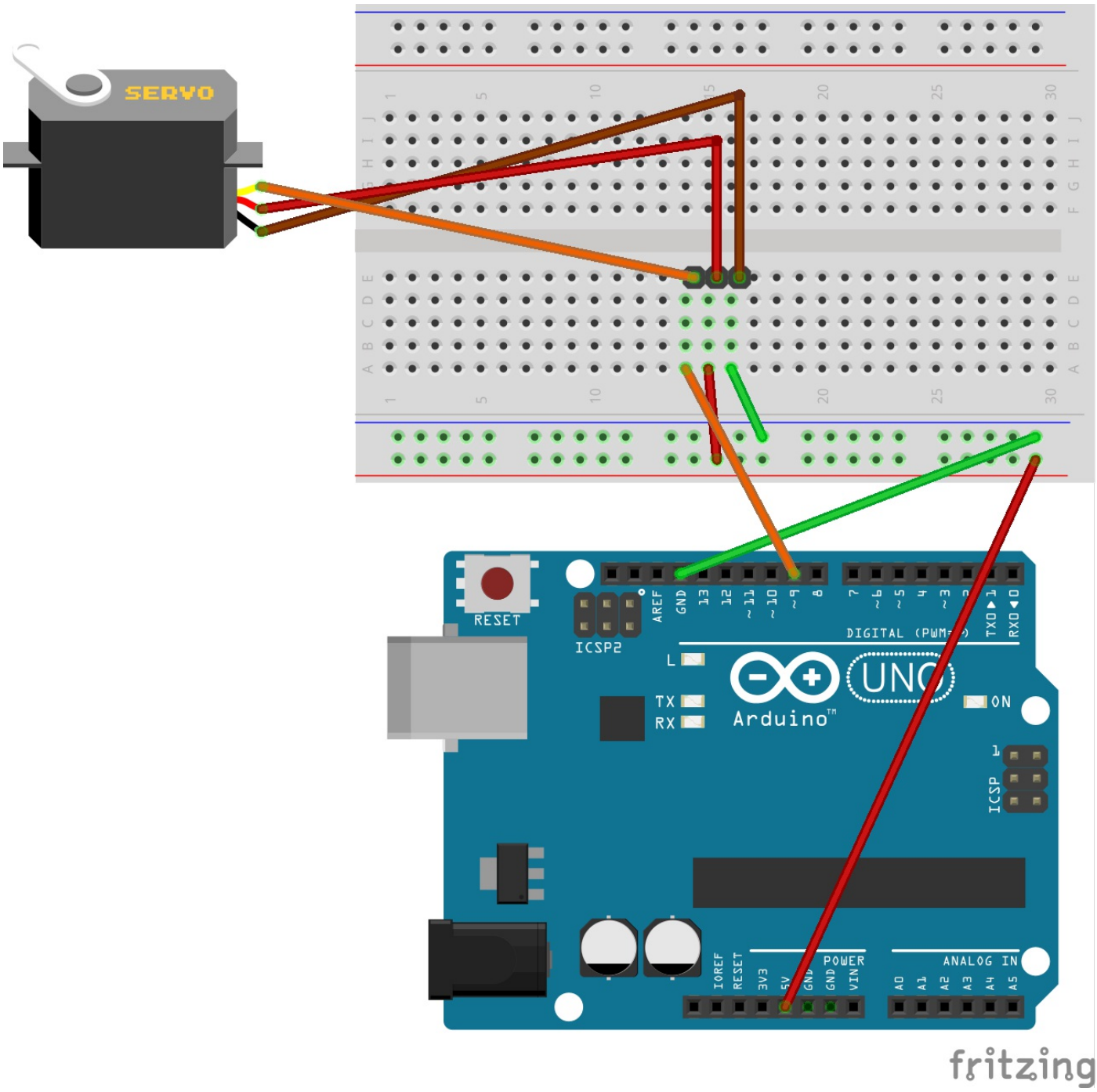
Supplies:

- Uno board and breadboard
- Servo motor
- Three jumper wires

Steps (Board):

The servo needs to be attached to an Arduino pin that is capable of **PWM**, since that's how the servo is powered. These are the pins with the ~ symbol, such as **~9**.

Refer to this Fritzing diagram:



1. Once your supplies are gathered, connect the breadboard to the Arduino:
    - *- rail* on the breadboard connects to *GND* on Uno
    - *+ rail* on the breadboard connects to *5V* on Uno
2. Next, connect the servo:
    i. First, refresh yourself on the polarity of the servo. Remember which jumper wires you attached to the servo wires. Remember that brown: ground, red: power, orange: data/signal.
    ii. Connect the jumper wire attached to the **brown servo wire (ground)** to the **GND (-)** rail on the breadboard.

iii. Connect the jumper wire attached to the **red servo wire (5V)** to the **power (+)** rail on the breadboard.
iv. Finally, connect the jumper wire attached to the **orange servo wire (signal)** to a jumper that connects to **pin 9** on the Arduino.

3. Note that if you change the pin number in either the board or the code, it must match. **You need to use a pin with a ~**.

Steps (Code):

Now for the code! You'll be getting the basic code template from a GitHub Gist account. There will be comments in the code for you to follow and modify.

1. Make sure to connect the Arduino to your computer and configure the board and port connection before moving on.
2. Navigate to this link and grab the sketch code from this Gist on Jonathan's GitHub:

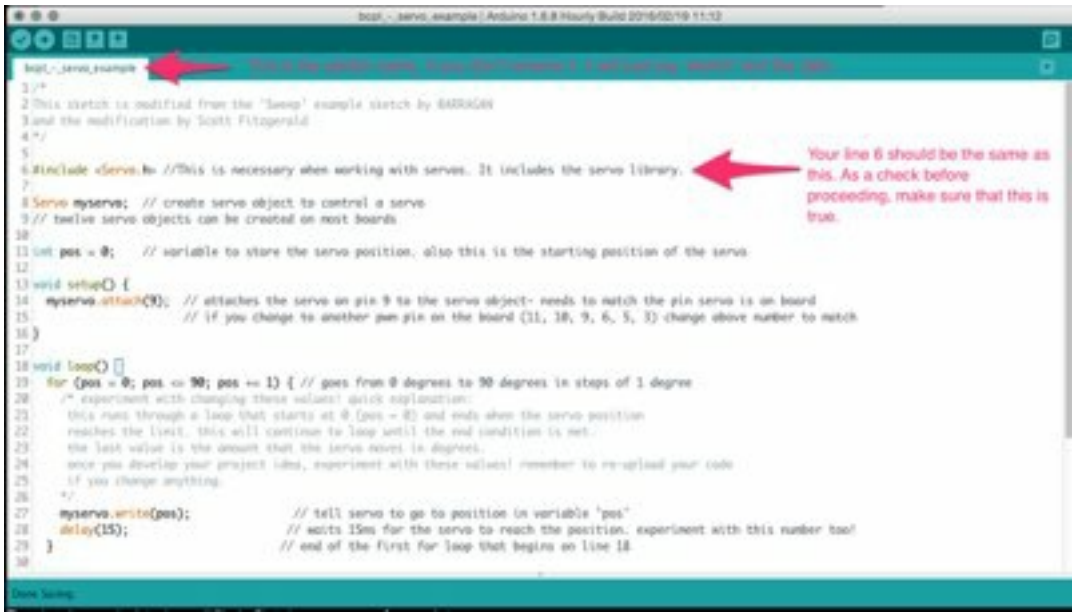
1. Copy the entire gist code Gist is a great way to share code with your learners. For this project, go ahead and copy the entirety of the code

(lines 1 - 41).

To copy, use **CMD + C** on a Mac, or **Ctrl + C** on Windows.

Note: Don't close the browser window incase you need to repeat this process!

1. Create a new Arduino sketch



2. Paste the code into the new sketch Once you've created the new sketch, paste the **entire** copied code into the sketch. You should replace everything- don't leave the original **setup and loop** functions or the code won't work. Check that your code matches the image before proceeding.

3. Begin modifying the code! Follow the comments and experiment. *Save your sketch!*

Going Further:

The modifications that you make depend on what you want to do for your project. This guide is for the basic servo example.

The code comments contain some suggestions for places to modify, but here are some starting points:

In the **void loop()** function, you can change the speed, starting position, and end position of the servo. If you want to do this, find the **for loops** on lines 19 and 31. The first one is for the forward motion of the servo and the second one is for the return motion. Experiment with modifying these positions and the **delay** values! Use the commented code as a reference point.