

## What Will You Learn?

Learn to connect and write code to control two Servos.

## Why Should You Learn This?

This lesson will walk you through how to wire two Servos to the Arduino and program for different Servo movements. Getting two servos to work together can really open up your options for creating new projects. Think, for example, about a robot that can nod and shake its head! You should have completed the "Servo" activity before beginning this activity.

## Here's What You'll Need:

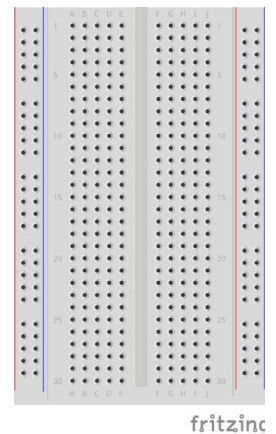
### Arduino Microcontroller



### 2 Servos



### Breadboard



### 3 Red Wires



### 3 Black Wires



### 2 White Wires

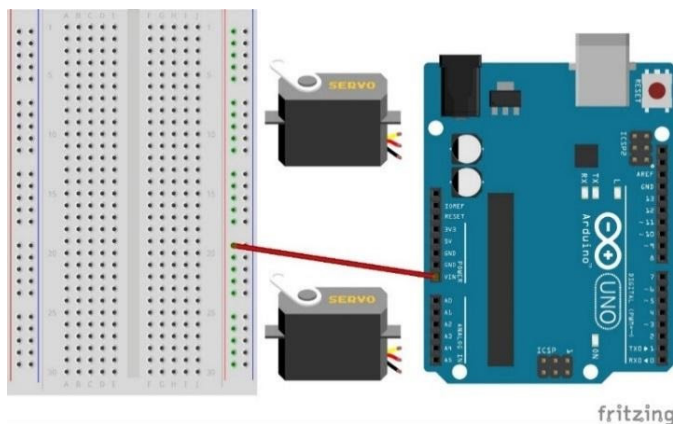


# Safety First!

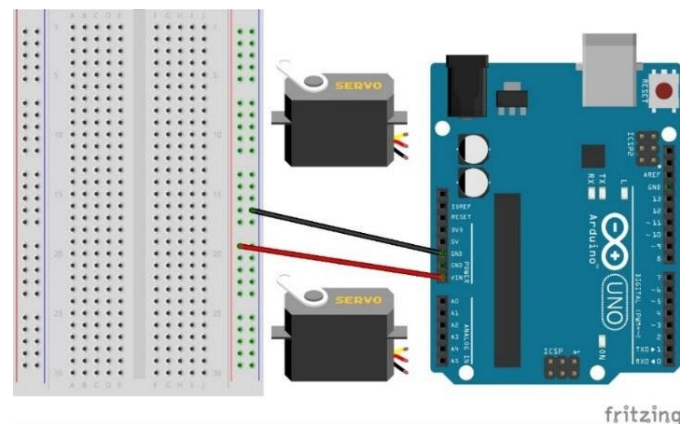
Never allow the red and black wires to touch together while they are connected to a powered microcontroller, as that creates a short circuit. A short circuit can potentially cause the wires and/or the microcontroller to get hot enough to burn the skin. In addition, in the event of a short circuit the microcontroller can potentially catch on fire.

## Let's Get Connected!

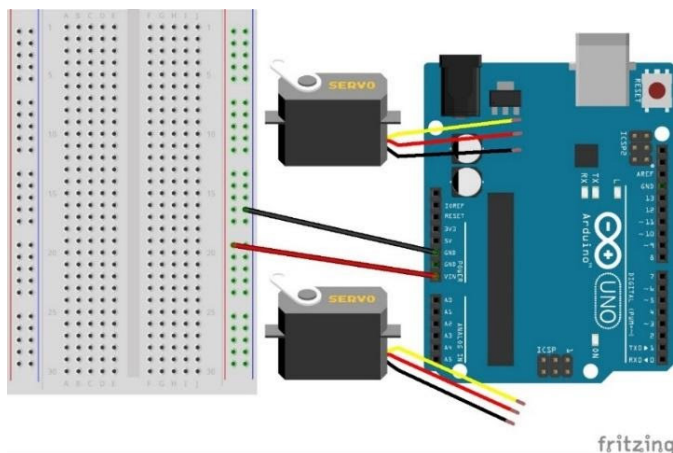
1. Start by connecting one of the red wires from Vin (+) to the positive rail, the red column, of the breadboard. This makes that entire column positive.



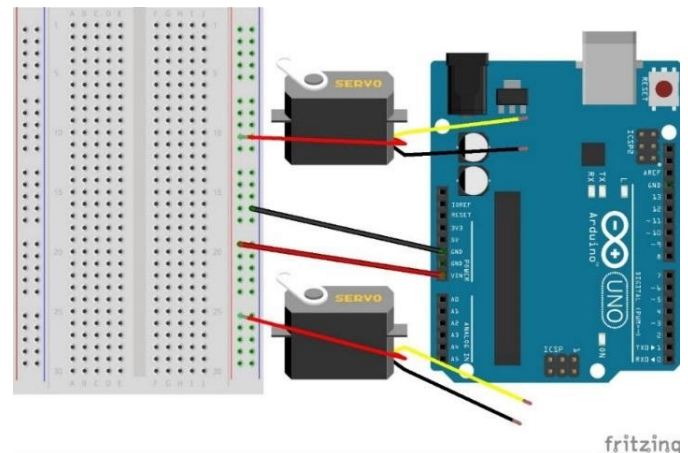
2. Next, connect one of the black wires from GND (-) to the negative rail, the blue column, of the breadboard. This makes that entire column negative.



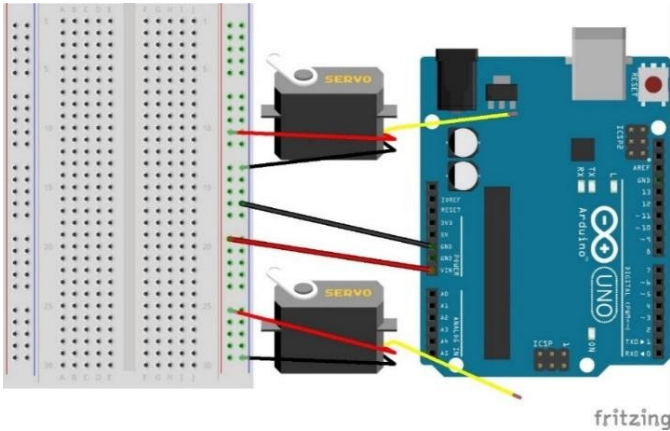
3. Connect the remaining red and black wires to their respective places on the Servos. Connect a white wire to each of the yellow Servo wires.



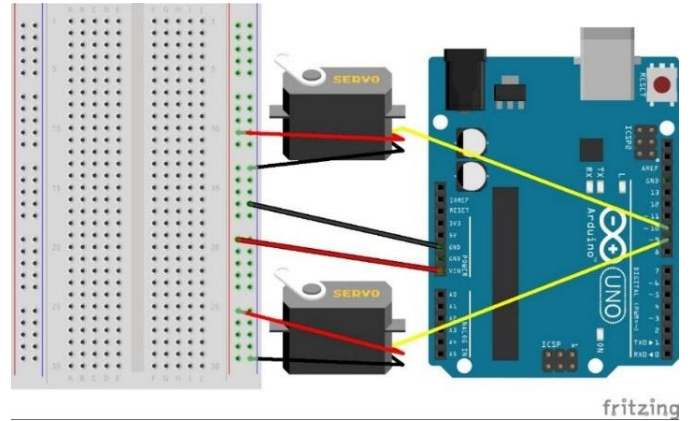
4. Connect both red wires from the Servos to the positive rail, the red column.



5. Connect both black wires from the Servos to the negative rail, the blue column.

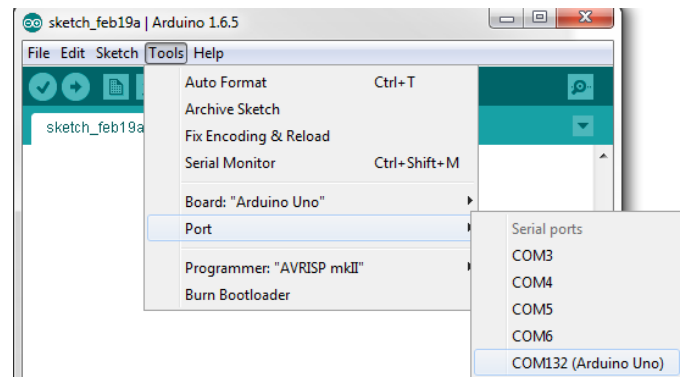
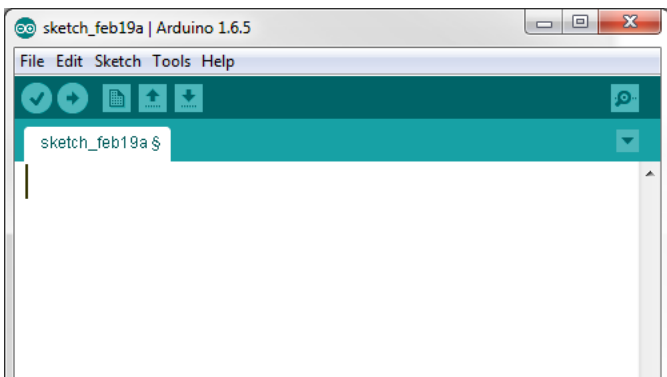


6. Connect the white (control) wire from one Servo to 9 and the other to 10.

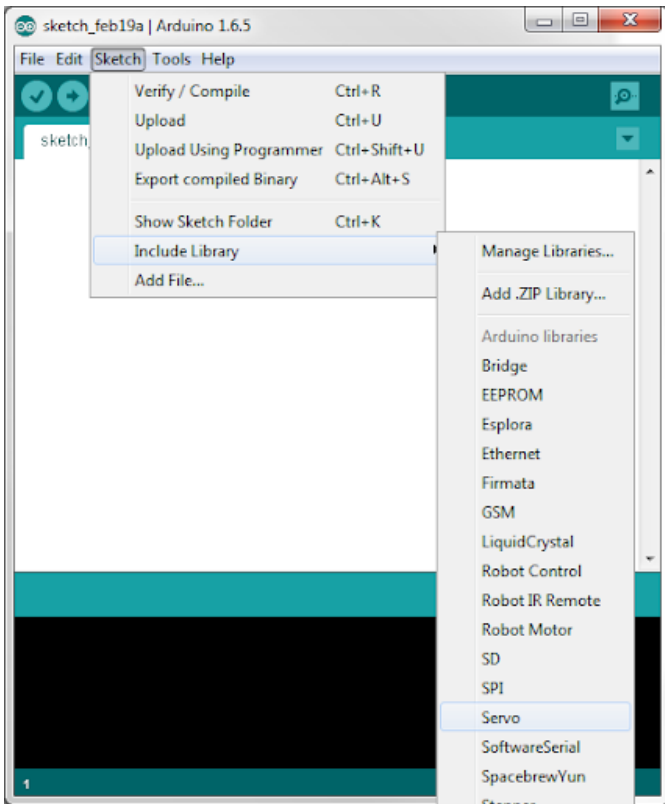


## Time to Write Some Code!

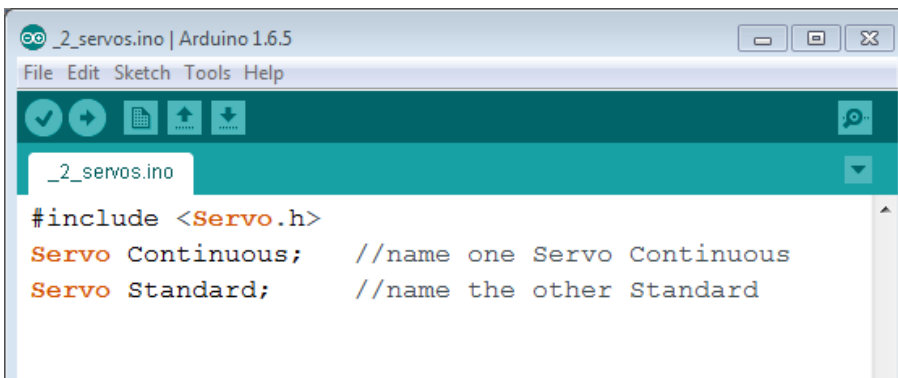
1. Start by connecting the Arduino microcontroller to the computer via the USB cable. Open a new sketch and delete all the text. Also, don't forget to check if you're connected to the right port.



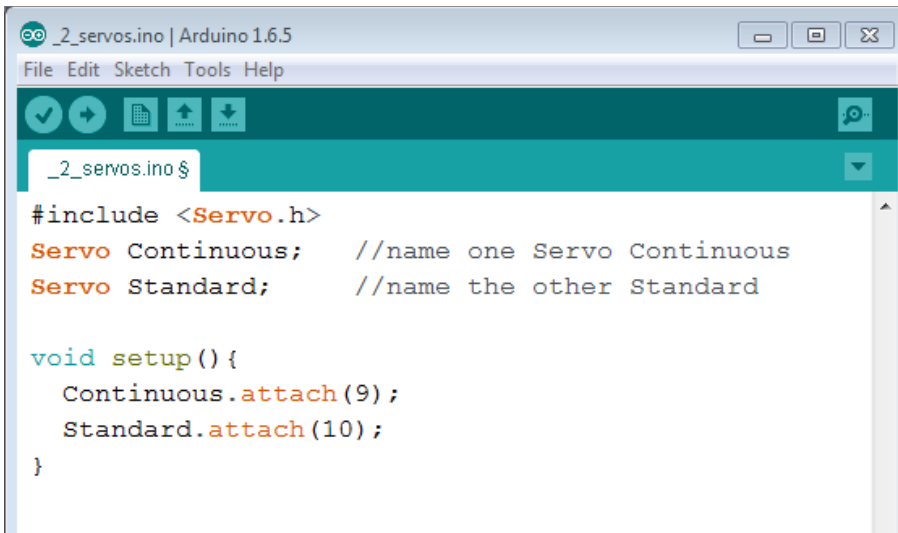
2. To create the signal needed to control the Servos, you will need to include the Servo library. To do this, go to Sketch → Include Library → and click on Servo. This brings up `#include <Servo.h>`. Even though you will be controlling 2 Servos, you only need to include the Servo library once.



3. Next, you'll need to declare and name the Servos. Since there are 2, they each need a unique name. For example: "Continuous" and "Standard", since you're using both types of Servos, but you can call them whatever you want.



4. Next is the setup function. Since there are 2 Servos, you will need 2 lines of code in the setup function in order to attach the Servos to the 2 different pins. Reminder: Servos always need to be attached to the digital pins on the microcontroller with the ~ next to them. In this activity, you will program the continuous Servo to 9 and the standard Servo to 10, so you need to make sure that the white wire of the continuous Servo is connected to pin 9 and the white wire of the standard Servo is connected to pin 10 on the microcontroller.

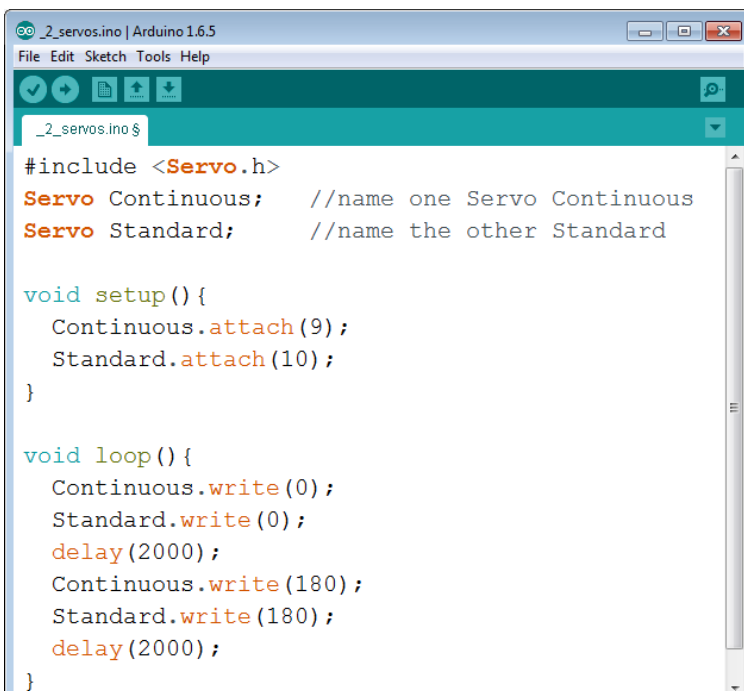


```
_2_servos.ino | Arduino 1.6.5
File Edit Sketch Tools Help
_ino $
#include <Servo.h>
Servo Continuous; //name one Servo Continuous
Servo Standard; //name the other Standard

void setup() {
  Continuous.attach(9);
  Standard.attach(10);
}
```

5. Now, it is time for you to get to the loop function. The following code is for basic motions for both Servos. Remember from the “Servo” activity that the same line of code works for both types of Servos (continuous and standard), but depending on their type, they will behave differently.

- a. First, use the value 0 for both Servos “Continuous.write(0);” and “Standard.write(0);”. This will cause the continuous Servo to go as fast as it can in one direction and the standard Servo to go to position 0 degrees.
- b. Next, put a delay so that the microcontroller will delay, or wait, before reading the next line of code and doing a different motion. A delay of 2000 means that the microcontroller will wait 2000 milliseconds (2 seconds) before reading the next line of code.
- c. Next, use the value 180 for both Servos by typing the lines “Continuous.write(180);” and “Standard.write(180);”. This will cause the continuous Servo to spin in the opposite direction as fast as it can and the standard Servo to go to position 180 degrees.
- d. Another delay of 2000 will again program the microcontroller to wait for 2 seconds before reading the next line of code and performing the next action. This means that the continuous Servo will keep spinning and the standard Servo will remain at 180 degrees for 2 seconds before the loop repeats.

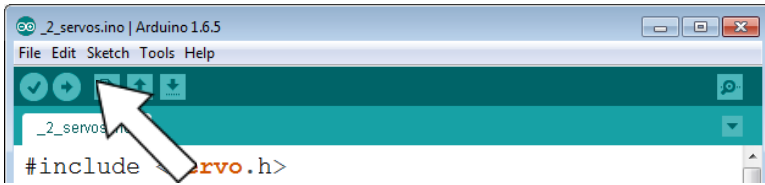


```
_2_servos.ino | Arduino 1.6.5
File Edit Sketch Tools Help
_ino $
#include <Servo.h>
Servo Continuous; //name one Servo Continuous
Servo Standard; //name the other Standard

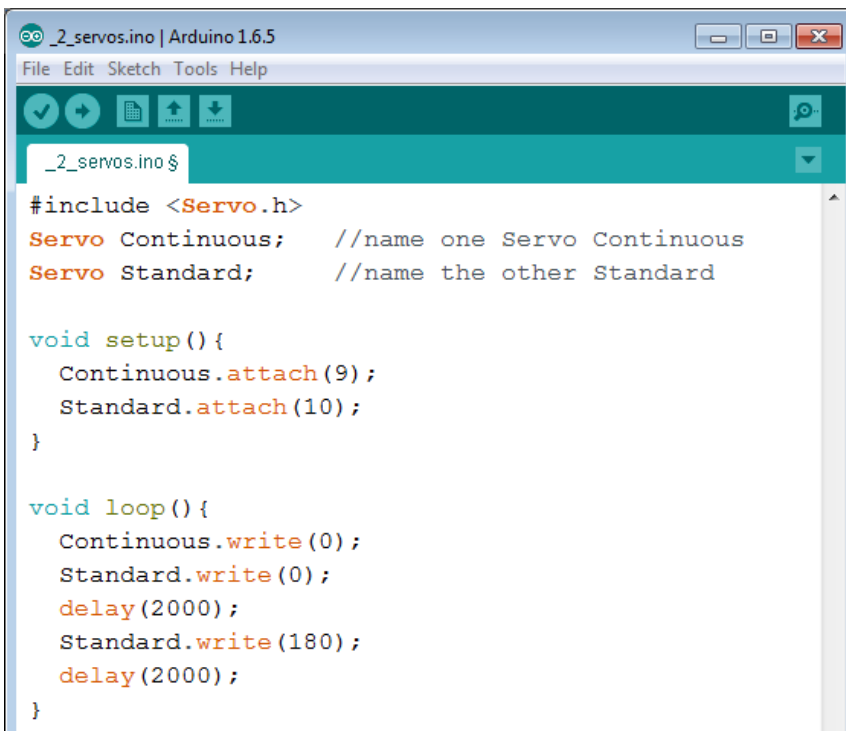
void setup() {
  Continuous.attach(9);
  Standard.attach(10);
}

void loop() {
  Continuous.write(0);
  Standard.write(0);
  delay(2000);
  Continuous.write(180);
  Standard.write(180);
  delay(2000);
}
```

6. Upload this code and see how your 2 Servos behave. You should see the continuous Servo spinning one way and then the other, changing direction every 2 seconds. You should see the standard Servo moving back and forth from 0 to 180 degrees and remaining at each stop for 2 seconds.



7. By making one change to the code above, the continuous Servo will spin in only one direction while the standard Servo rotates between 0 and 180 degrees. Focus on the code for the continuous Servo. Before the delay, the continuous Servo was set to 0 (`Continuous.write(0);`) and then later it was set to 180 (`Continuous.write(180);`). There is a delay between those two motions so that the Servos have time to move before they are sent new commands. However, if you remove one of the "write" commands for the continuous Servo, it will only spin in one direction. In this example, changing the delay values would only affect the standard Servo, but the continuous Servo is still only getting one command, so it only spins in one direction.



# Tips for Success!

- If you're having errors uploading, check that you have written everything correctly and that you are connected to the right port.
- If your code uploaded, but the Servos are not moving, you may have an issue with your wiring. Try pulling everything out and starting over. Sometimes it's difficult to see what the problem is until you start all over.
- If your wiring looks good and your code is fine, try each Servo, one at a time, to see if it is an issue with the Servo itself.

## Are You Ready for Some Challenges?

1. To really understand how a delay affects the Servos, can you write a program that makes the standard Servo go from 0 to 180 degrees with a delay of 200, while the continuous Servo changes directions every 1 second?

**Hint:** Remember, the delay programs the microcontroller to know how long to wait before reading the next line of code and it always reads through the code in order. When the code calls for a delay, it waits however long you program it before reading the next line of code. So, anything you told it to do before that delay it is going to continue doing.

Take, for example, the code written below. When we focus on the standard Servo, we see that each write command is followed by a delay of 200, so the Servo will move between 0 and 180 degrees with a delay of 200 milliseconds. However, when we focus on the continuous Servo, there are 4 delays of 200 between the first command "Continuous.write(0);" and the second command "Continuous.write(180);". This means that the Servo will spin in one direction for 800 milliseconds and then in the opposite direction for 800 milliseconds.

Full Code in "void loop(0) {"	Focus on standard Servo	Focus on continuous Servo
<pre>Continuous.write(0); Standard.write(0); delay(200); Standard.write(180); delay(200); Standard.write(0); delay(200); Standard.write(180); delay(200); Continuous.write(180); Standard.write(0); delay(200); Standard.write(180); delay(200); Standard.write(0); delay(200); Standard.write(180); delay(200);</pre>	<pre>Continuous.write(0); Standard.write(0); delay(200); Standard.write(180); delay(200); Standard.write(0); delay(200); Standard.write(180); delay(200); Continuous.write(180); Standard.write(0); delay(200); Standard.write(180); delay(200); Standard.write(0); delay(200); Standard.write(180); delay(200);</pre>	<pre>Continuous.write(0); Standard.write(0); delay(200); Standard.write(180); delay(200); Standard.write(0); delay(200); Standard.write(180); delay(200); Continuous.write(180); Standard.write(0); delay(200); Standard.write(180); delay(200); Standard.write(0); delay(200); Standard.write(180); delay(200);</pre>

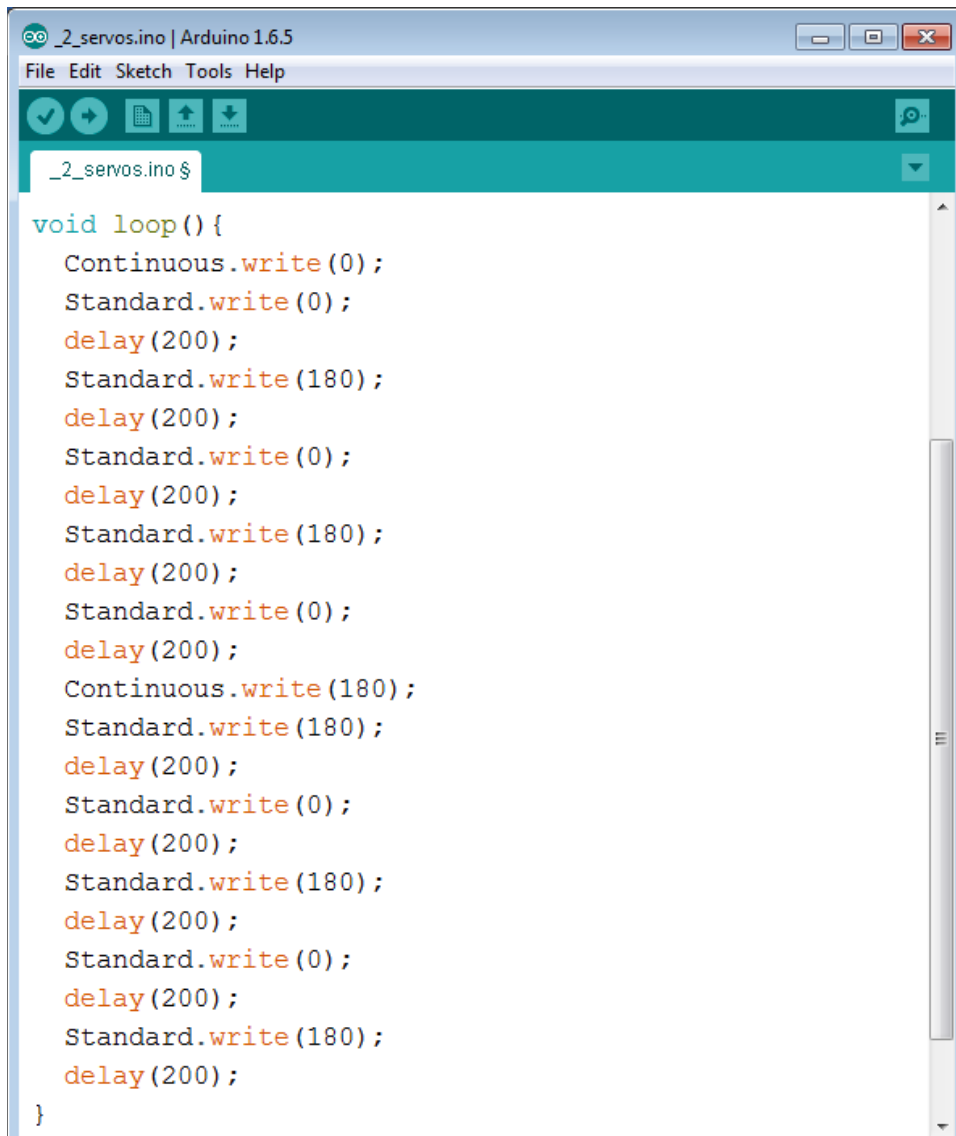
With this additional help, can you now write a program that makes the standard Servo go from 0 to 180 degrees with a delay of 200, while the continuous Servo changes directions every 2 seconds?

2. Can you write a program that controls 3 Servos? Write a program that controls 2 continuous Servos and 1 standard Servo. Program one of the continuous Servos to spin one direction and the other to spin the opposite direction, while the standard Servo moves between 0 and 90 degrees every second. Don't forget to connect the wires for this additional Servo!

## Challenge Solutions:

Challenge 1 Solution:

**Note:** Only the loop function of the code is shown as the setup and pre-setup remained the same.

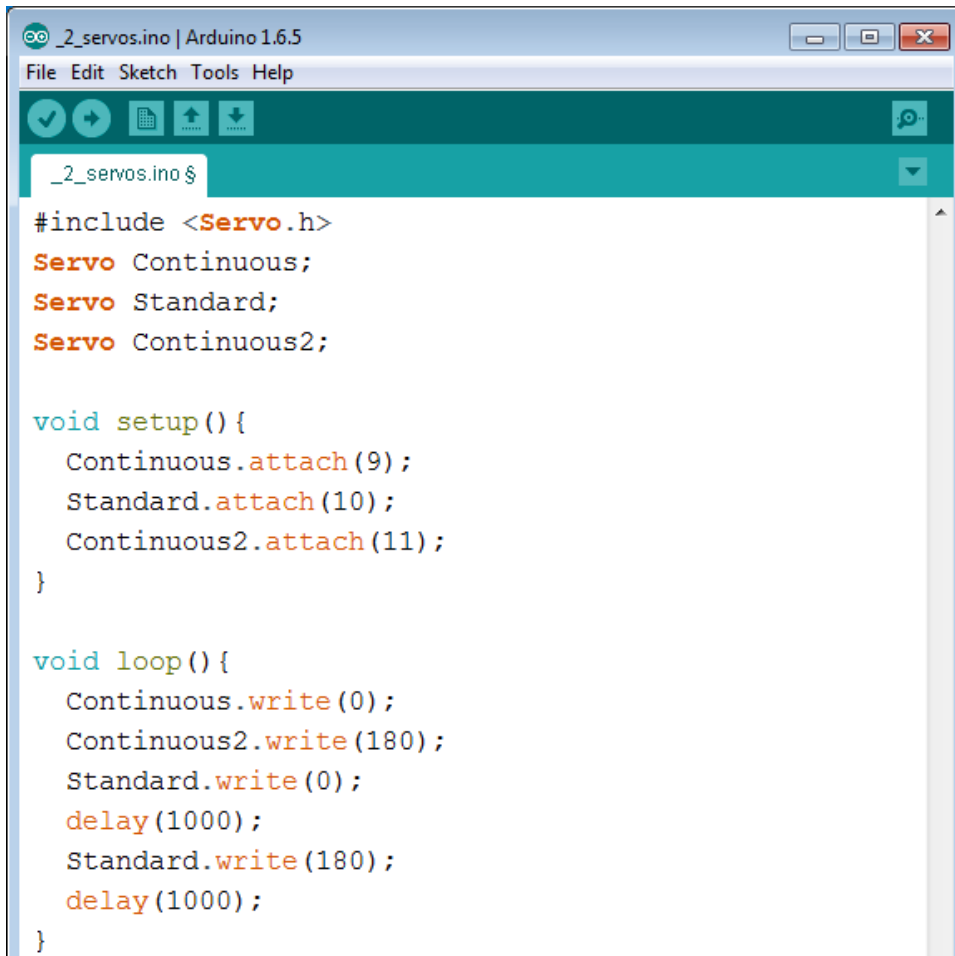
A screenshot of the Arduino IDE interface. The window title is "\_2\_servos.ino | Arduino 1.6.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar shows icons for running, uploading, and other IDE functions. The main editor area displays the following C++ code:

```
void loop() {  
  Continuous.write(0);  
  Standard.write(0);  
  delay(200);  
  Standard.write(180);  
  delay(200);  
  Standard.write(0);  
  delay(200);  
  Standard.write(180);  
  delay(200);  
  Standard.write(0);  
  delay(200);  
  Continuous.write(180);  
  Standard.write(180);  
  delay(200);  
  Standard.write(0);  
  delay(200);  
  Standard.write(180);  
  delay(200);  
  Standard.write(0);  
  delay(200);  
  Standard.write(180);  
  delay(200);  
}
```



## Challenge 2 Solution:

Notice that you have to declare the new Servo in your pre-setup and attach the new Servo in your setup function.



```
_2_servos.ino | Arduino 1.6.5
File Edit Sketch Tools Help
_2_servos.ino $
#include <Servo.h>
Servo Continuous;
Servo Standard;
Servo Continuous2;

void setup() {
  Continuous.attach(9);
  Standard.attach(10);
  Continuous2.attach(11);
}

void loop() {
  Continuous.write(0);
  Continuous2.write(180);
  Standard.write(0);
  delay(1000);
  Standard.write(180);
  delay(1000);
}
```