Qualcomm
thinkabit lab

What Will You Learn?

Learn to wire and write code to control a servo and an LED at the same time.

Why Should You Learn This?

Now that you know how to use a LED and a servo separately, you can expand on your ideas by learning how to program both together. While this is more complicated than wiring each of the components separately, it is something that is possible with a bit of time and a few extra parts.

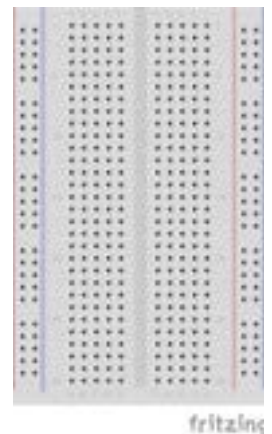# Here's What You'll Need:

Arduino Microcontroller

1 Servo

Continuous servo is used in this activity, but you could also use a standard Servo)

Breadboard

2 Red Wires

2 Black Wires

1 White Wire

**1 resistor (100 – 300 ohms)**

1 LED

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. Let's focus on connecting the servo first. Begin by connecting a red wire from Vin on the microcontroller to the red (+) wire of the servo.
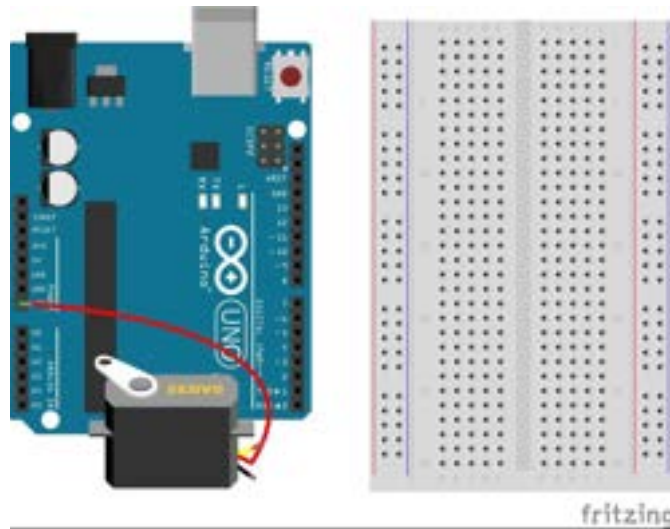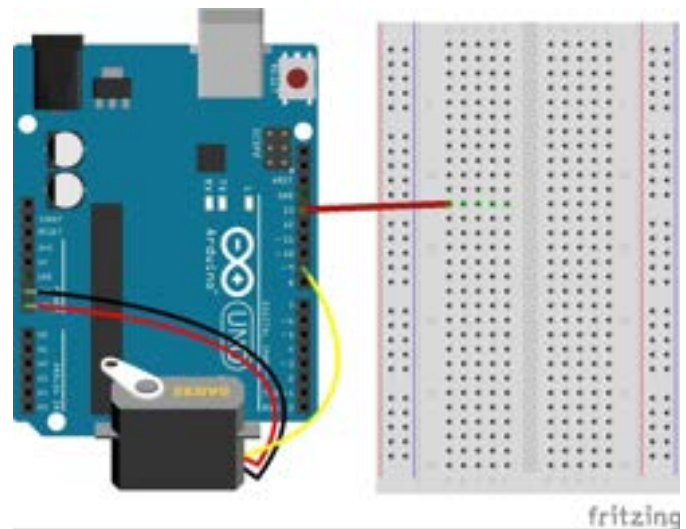

2. Next, connect a black wire from GND to the black (-) wire of the servo.
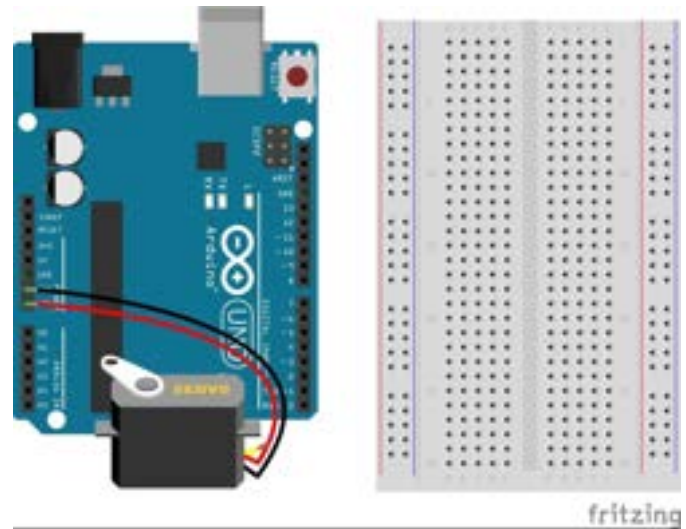

3. To finish hooking up the servo, connect a white wire from 9 to the white (control) wire of the servo (the wire in the diagram is yellow, but your wire may be white).
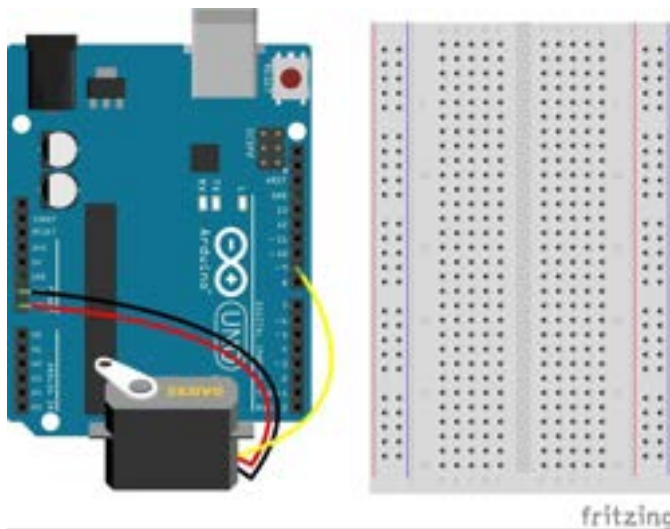

4. Now that the servo is connected, you can focus on the LED. Connect a red wire from 13 to any row of the breadboard (row 12 is used below). You will use this same row to connect your resistor later.

5. Next, connect a black wire from GND to the negative rail, or the blue lined column.
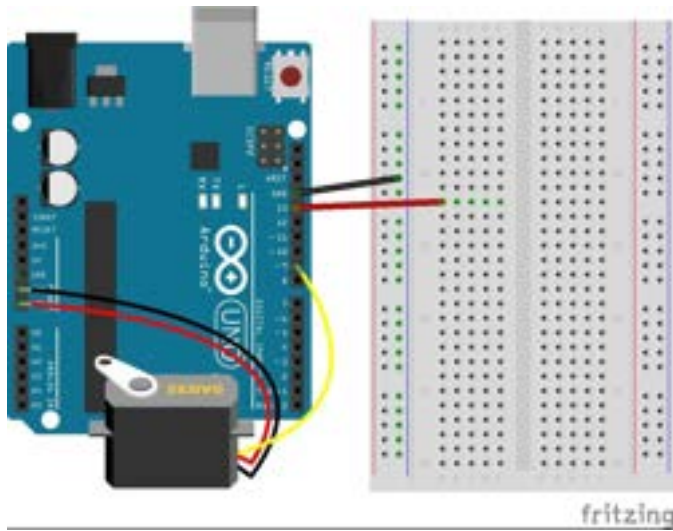
6. You will also need a resistor to make sure you don't burn out your LED. Connect the resistor from the row that has the red wire (row 12 below) to another row on the breadboard (row 8 is used below).
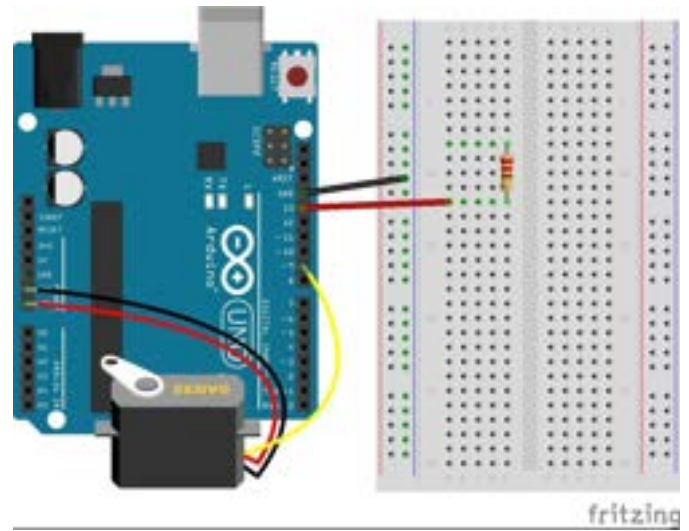
7. Lastly, connect the LED. The long leg (positive) will go in the same row as the end of the resistor that is not connected to the red wire (row 8), and the short leg (negative) will go in the negative rail.
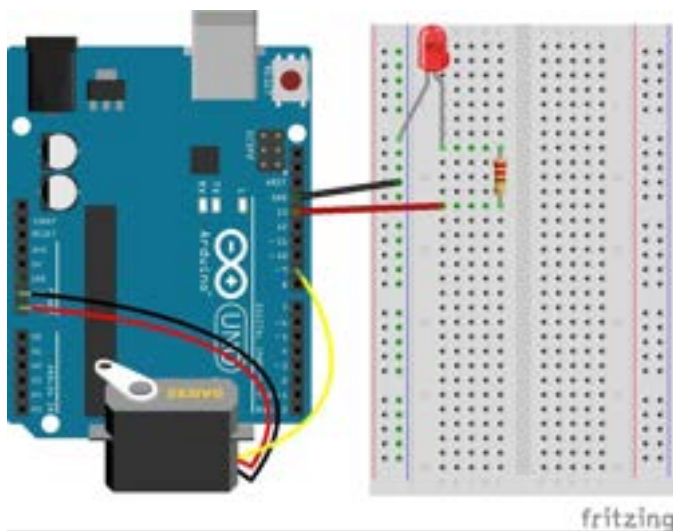
Time to Write Some Code
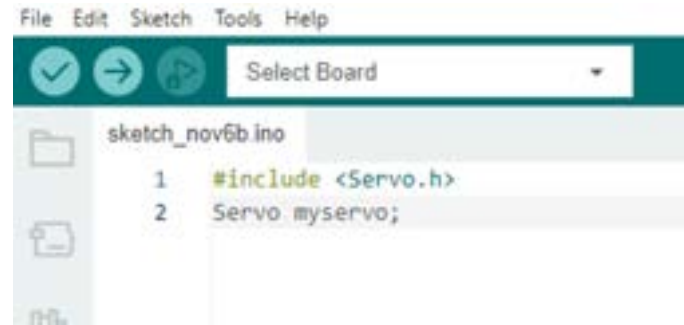
1. Start by opening the Arduino programming language and deleting the text to start with a blank document. This is a good time to make sure that you've selected the right port to connect your board to by choosing the COM that is listed under Arduino Uno. It may be tempting to j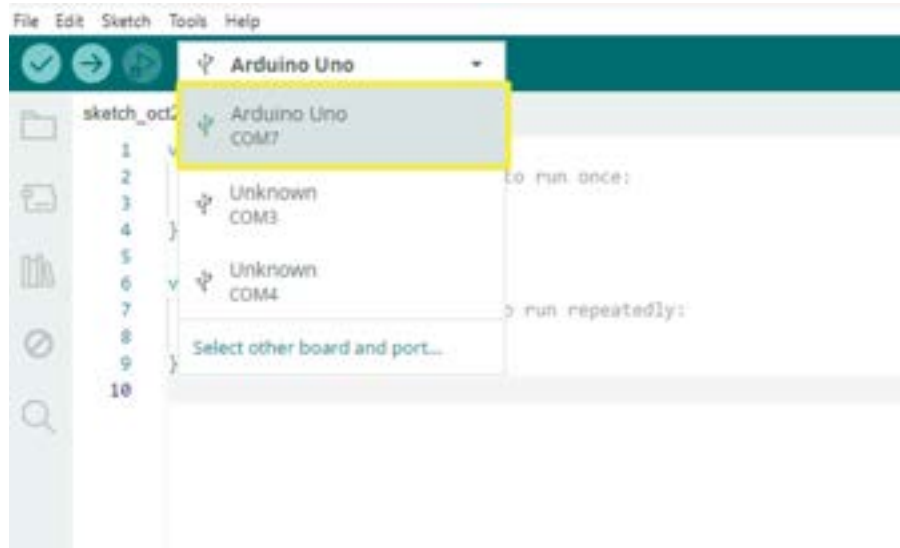ust copy and paste the Servo code and LED Blink code into one document, but combining the two programs directly won't work. In this activity, you'll learn how to write the code from start to finish.

2. Whenever you use a servo, you need to include the servo library in your program. If you haven't downloaded the servo library, go back to the Servo activity to learn how. To add that library to your sketch, go to Sketch -> Include Library -> Servo in the menu. "**#include** **<Servo.h>**" will appear in your sketch. You can also just type this in to add the servo library.
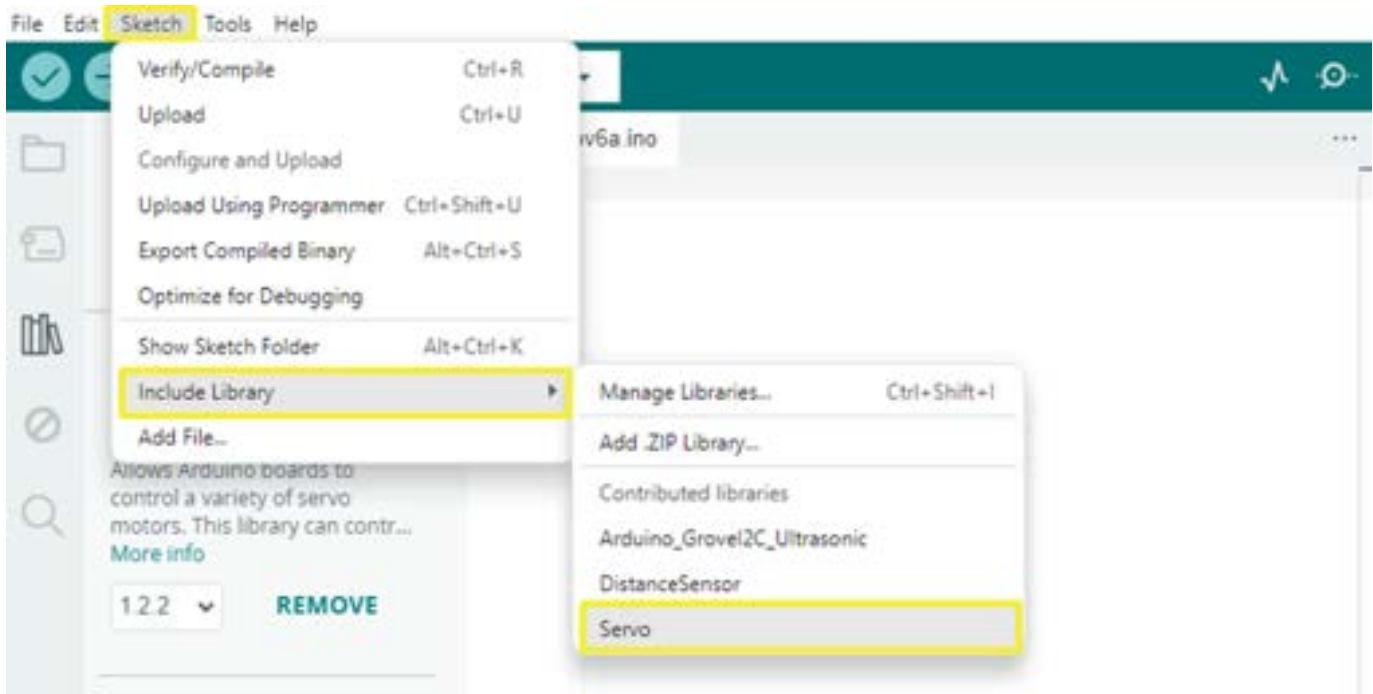
3. Next, you need to name your servo, as you did in the "Servo" activity. The example below names the servo "myservo". You can name your servo whatever you want, just be sure to use that name throughout your entire code.

File Edit Sketch Tools Help

Select Board

sketch_nov6b.ino
```
1   #include <Servo.h>
2   Servo myservo;
```

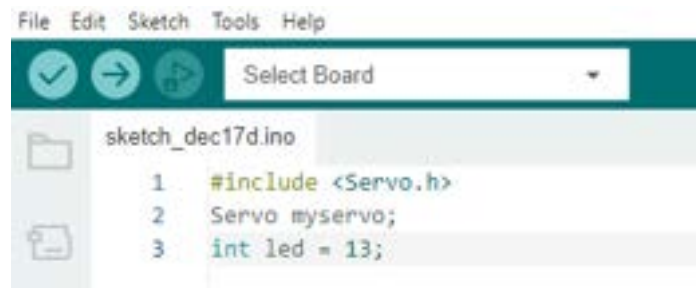4. You are also going to make a name for your LED. To do this, you need to create a variable that tells you what pin your LED is connected to. To create a variable, use the line of code "**int led=13**". In this line of code, "led" is the name of your variable (it's good to make this name something easy to understand). 13 is the pin that your LED is connected to, and "int" is the type of variable that you are creating. "int" is short for integer, which just means that it will be a number. This is the most common type of variable that you will use.

File Edit Sketch Tools Help

Select Board

sketch_dec17d.ino
```
1   #include <Servo.h>
2   Servo myservo;
3   int led = 13;
```

5. Now that you have named your servo and LED, you're ready to write the Setup section of your program. Because you are using 2 components, a servo and an LED, you need to setup both. You'll set up the servo and the LED the same way that you did in the "Servo" and "LED Blink" activities, but this time you will put them both in the same program.

File Edit Sketch Tools Help

Select Board

LED_Servo_FY25.ino
```
1   #include <Servo.h>
2   Servo myservo;
3   int led = 13;
4
5   void setup(){
6     myservo.attach(9);
7     pinMode(led, OUTPUT);
8   }
```
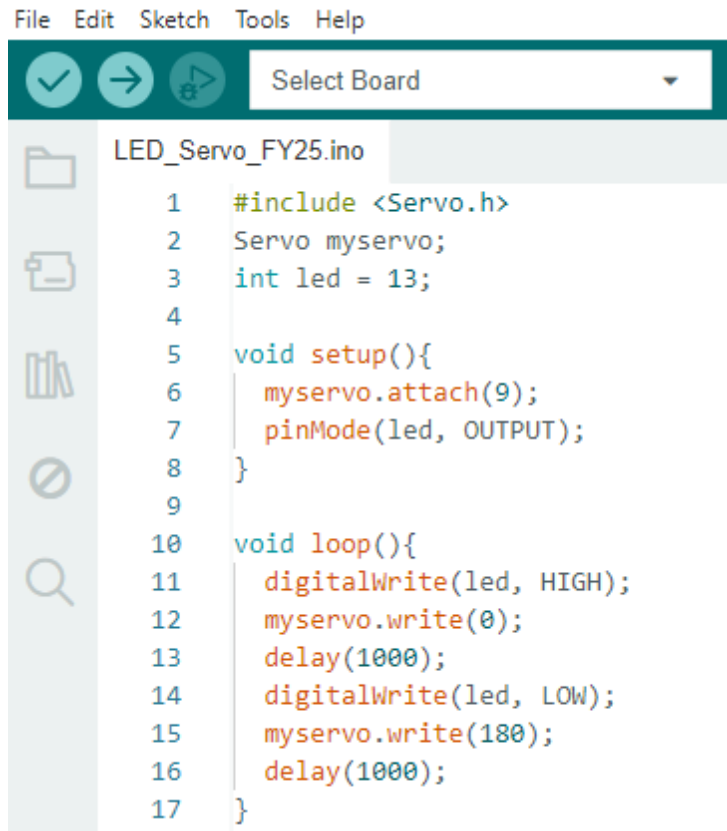
6. Next, you can write the loop function of your program. It may be tempting to copy and paste the code from the "Servo" and "LED Blink" examples together. If you tried that you would end up with the following:

```
10    void loop(){
11      digitalWrite(led, HIGH);   // turn the LED on
12      delay(1000);               // wait one second
13      digitalWrite(led, LOW);    // turn the LED off
14      delay(1000);               // wait one second
15
16      myservo.write(0);          // spin the servo to the left
17      delay(2000);
18      myservo.write(180);        // spin the servo to the right
19      delay(2000);
20    }
```

Although this will work, it will not make the LED and servo do what you want them to do – which is to have the LED blink and servo move at the same time. The LED will turn on, stay on for a second, then turn off, but because the microcontroller runs through all the lines of code one at a time before starting over from the top, the light will stay off for 5 seconds (1000 + 2000 + 2000). The same thing happens with the servo. Since you are using a continuous servo, it will start out not moving, then will spin one direction for 2 seconds and then in the opposite direction for 4 seconds (2000 + 1000 + 1000). This table may help you see it more clearly:

Full Code	What each line of code does	What we see when the code runs
| ```
10    void loop(){
11      digitalWrite(led, HIGH);   // turn the LED on
12      delay(1000);               // wait one second
13      digitalWrite(led, LOW);    // turn the LED off
14      delay(1000);               // wait one second
15
16      myservo.write(0);          // spin the servo to the left
17      delay(2000);
18      myservo.write(180);        // spin the servo to the right
19      delay(2000);
20    }
``` | | LED turns on for 1 second

LED turns off for 5 seconds (will stay off until the next loop of code)

servo spins to the left for 2 seconds

servo spins right for 4 seconds (until the servo turns right in the next loop) |

If you want the light to blink on and off every second and the servo to change directions every second, you will need to do both actions (for the LED and the servo) before each delay. It should look like the code below:

File   Edit   Sketch   Tools   Help

Select Board

LED_Servo_FY25.ino

```
1    #include <Servo.h>
2    Servo myservo;
3    int led = 13;
4
5    void setup(){
6      myservo.attach(9);
7      pinMode(led, OUTPUT);
8    }
9
10   void loop(){
11     digitalWrite(led, HIGH);
12     myservo.write(0);
13     delay(1000);
14     digitalWrite(led, LOW);
15     myservo.write(180);
16     delay(1000);
17   }
```

Congratulations. You should now have an LED blinking and a servo spinning back and forth in unison. Feel free to change the delay or make the servo spin slower by changing the value in **myservo.write( );** (see "Programming a Servo" for more information).

Tips for Success

- If you're having issues with your code, review all the lines in the example code to make sure you didn't miss something or refer to the "Troubleshooting" document for more help.

- If your LED is not working, be sure that all your connections have been made properly and the LED is plugged in the right way. If you've checked each of these things, then try a different LED.

- If your servo is not working, make sure that all your wires are in the correct pins and that the red, black and white wires are in the right order on the servo attachment. If the wiring looks correct, you may want to try another servo.

Are You Ready for Some Challenges?

Below are some other combinations of blinking and spinning for you to try as challenge problems. The possible solutions are posted below, but try doing them yourself first. Remember, code can look different and have the same effect.

1. Try to make your LED continue to blink at the same rate (on for 1 second, then off for 1 second) and have your servo spin only one way (instead of both ways). This will allow you to change the rate of the blinking without changing the motion of the servo.

2. Try to make your LED Blink fast (with a delay of 100 milliseconds) and your servo spin both ways for 1 second each direction.

3. Try to incorporate the LED Fade code with a servo. You may want to start by having the servo only spinning one way and then try to have it change directions with the rate of the fade.

Challenge Solutions:

Note: For most of these challenges, I've only posted the loop, since the variables and setup are the same. I have also posted comments, as it's good practice to write comments with your code, but you do not need to write them in your code as it doesn't change what the code does. Remember, comments are for humans.

Challenge 1 Solution:

```
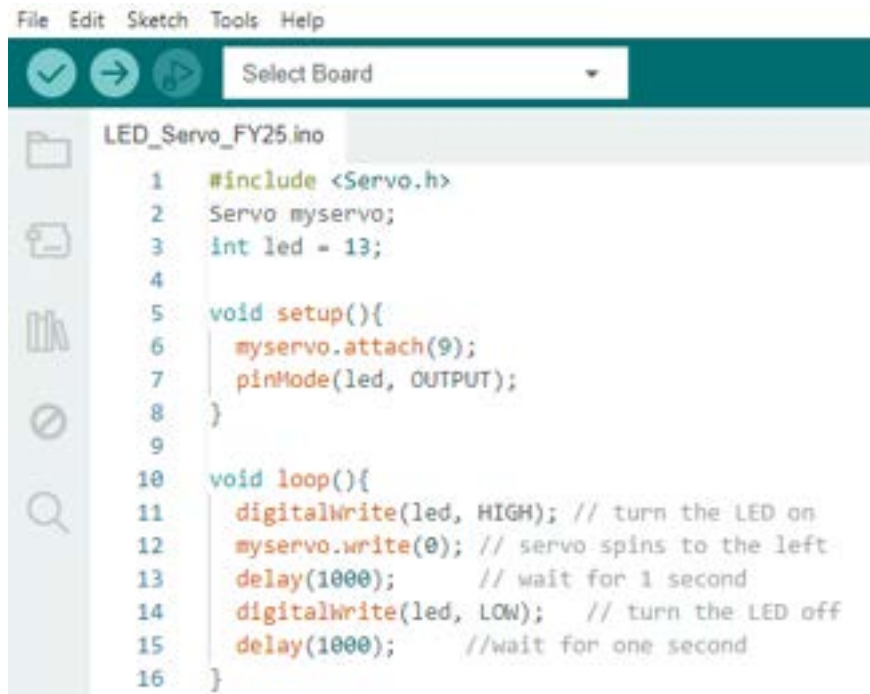File  Edit  Sketch  Tools  Help

Select Board

LED_Servo_FY25.ino
1    #include <Servo.h>
2    Servo myservo;
3    int led = 13;
4
5    void setup(){
6      myservo.attach(9);
7      pinMode(led, OUTPUT);
8    }
9
10   void loop(){
11     digitalWrite(led, HIGH); // turn the LED on
12     myservo.write(0); // servo spins to the left
13     delay(1000);       // wait for 1 second
14     digitalWrite(led, LOW);   // turn the LED off
15     delay(1000);      //wait for one second
16   }
```

You can always change the delay to be longer or shorter. This will change the rate at which your light blinks, but because you only told your servo to spin one direction and never changed the direction, it will continue to spin that direction no matter the value of the delay.

Challenge 2 Solution:

We want to have a longer delay for the servo than the LED, but we can't assign a separate delay to each part because a delay makes the entire program wait before it reads the next line. So, the loop in our program should look something like this:

```
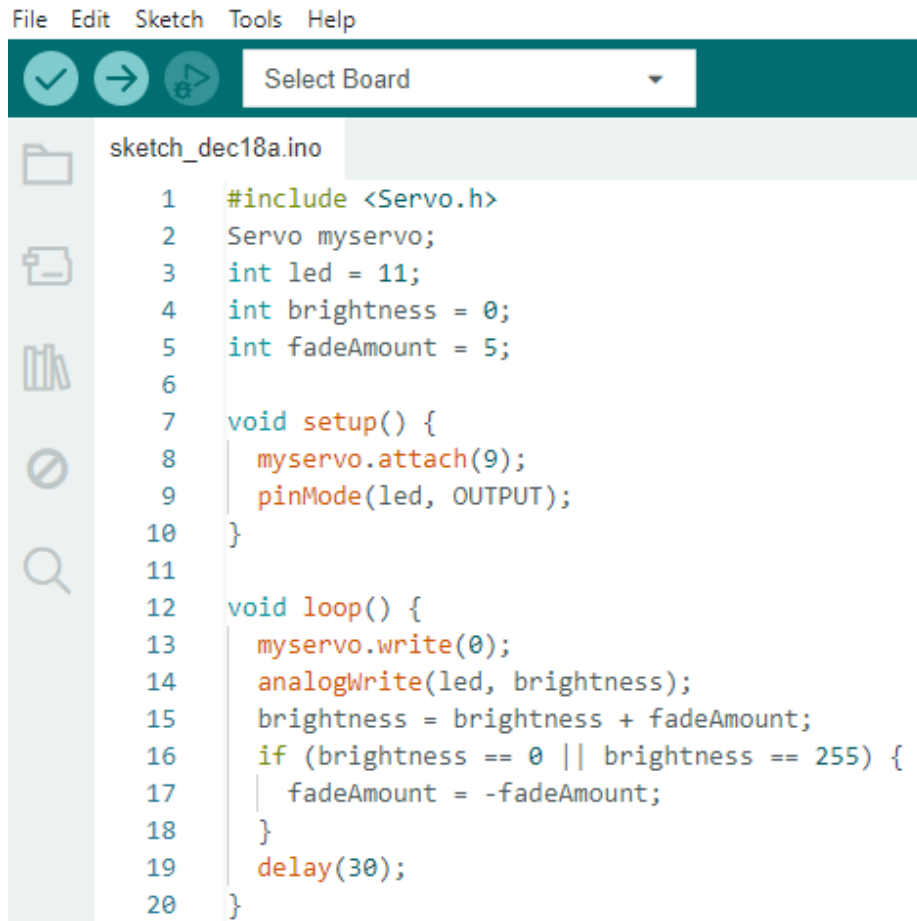File   Edit   Sketch   Tools   Help

   ✓   →   ⊕      Select Board              ▾

      LED_Servo_FY25.ino
  9
 10    void loop(){
 11      myservo.write(0); // servo spins to the left
 12      digitalWrite(led, HIGH); // turn the LED on
 13      delay(100);  //servo delay - 100
 14      digitalWrite(led, LOW);    // turn the LED off
 15      delay(100);  // servo delay - 200
 16      digitalWrite(led, HIGH); // turn the LED on
 17      delay(100);  // servo delay = 300
 18      digitalWrite(led, LOW);    // turn the LED off
 19      delay(100);  // servo delay = 400
 20      digitalWrite(led, HIGH); // turn the LED on
 21      delay(100);  // servo delay - 500
 22      digitalWrite(led, LOW);    // turn the LED off
 23      delay(100);  // servo delay - 600
 24      digitalWrite(led, HIGH); // turn the LED on
 25      delay(100);  // servo delay = 700
 26      digitalWrite(led, LOW);    // turn the LED off
 27      delay(100);  // servo delay = 800
 28      digitalWrite(led, HIGH); // turn the LED on
 29      delay(100);  // servo delay - 900
 30      digitalWrite(led, LOW);    // turn the LED off
 31      delay(100);  // servo delay - 1000
 32      myservo.write(180);  //servo spins to the right
 33      digitalWrite(led, HIGH); // turn the LED on
 34      delay(100);  // servo delay - 100
 35      digitalWrite(led, LOW);    // turn the LED off
 36      delay(100);  // servo delay = 100
 37      digitalWrite(led, HIGH); // turn the LED on
 38      delay(100);  // servo delay = 300
 39      digitalWrite(led, LOW);    // turn the LED off
 40      delay(100);  // servo delay - 400
 41      digitalWrite(led, HIGH); // turn the LED on
 42      delay(100);  // servo delay - 500
 43      digitalWrite(led, LOW);    // turn the LED off
 44      delay(100);  // servo delay - 600
 45      digitalWrite(led, HIGH); // turn the LED on
 46      delay(100);  // servo delay - 700
 47      digitalWrite(led, LOW);    // turn the LED off
 48      delay(100);  // servo delay = 800
 49      digitalWrite(led, HIGH); // turn the LED on
 50      delay(100);  // servo delay - 900
 51      digitalWrite(led, LOW);    // turn the LED off
 52      delay(100);  // servo delay - 1000
 53    }
```

Challenge 3 Solution:

I started with the LED Fade example and then added in the servo code to make the servo spin only one way. Making the servo change directions is a harder task, but may be easier once you have completed "Servo Sweep Challenge #2".

File Edit Sketch Tools Help

Select Board ▼

sketch_dec18a.ino

```arduino
1    #include <Servo.h>
2    Servo myservo;
3    int led = 11;
4    int brightness = 0;
5    int fadeAmount = 5;
6
7    void setup() {
8      myservo.attach(9);
9      pinMode(led, OUTPUT);
10   }
11
12   void loop() {
13     myservo.write(0);
14     analogWrite(led, brightness);
15     brightness = brightness + fadeAmount;
16     if (brightness == 0 || brightness == 255) {
17       fadeAmount = -fadeAmount;
18     }
19     delay(30);
20   }
```