The Bronx High School of Science
Ms. Thomas

Physical Science Department
Electrical Engineering

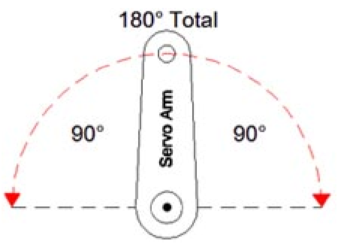
Names: ____ **Period:** ____________

Arduino Servo Motors

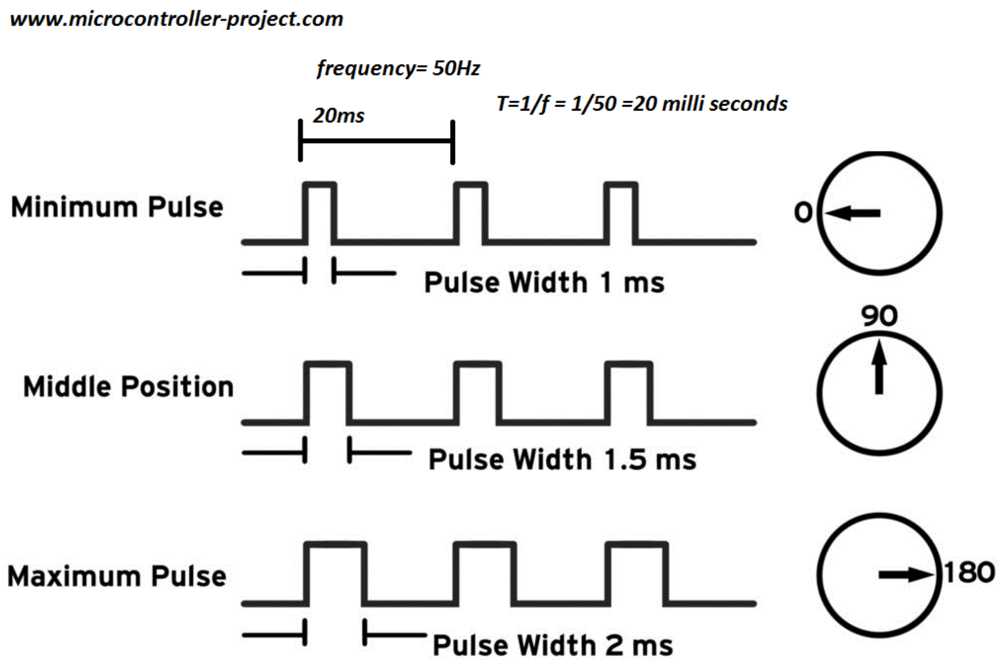
In this lesson we will learn how to make a servo motor (often just called a servo) spin to different positions.

Background

Servo motors, often just called servos, are devices used in robots to provide torque (a driving rotational force or "spin"). What distinguishes a servo from a regular DC motor is that a servo has its own microcontroller. This means a servo can be programmed to achieve a more precise output, which gives the user more control over it than a regular DC motor.

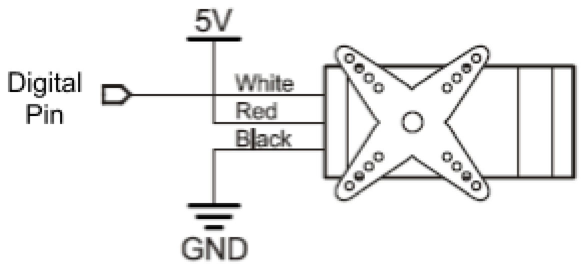


There are different types of servos. The type we will be using is known as a standard servo. The shaft of a normal standard servo is able to rotate 180º. The position or "angle" of the servo's shaft is controlled using something similar to a PWM signal. The width of the pulses sent set the shaft position/angle .



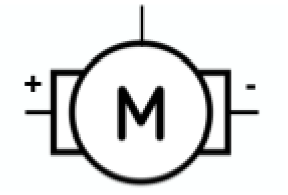
The process of programming an Arduino to control a servo is streamlined using the **Arduino Servo Library**, which comes with a number of useful functions that make controlling the servo easy. But be aware: using it also brings some additional complications (which you will explore later).

A servo has three wires connected to a set of three pins. The pin attached to the power wire (often red) must be connected to 5V. The pin attached to the ground wire (often black, but in our servos is brown) must be connected to 0V (GND). The pin attached to the signal wire (often white, but in our servos is orange) must be connected to a digital pin of the Arduino (a PWM-specific pin is not required).



Small servos often have an operating voltage of about 5V. The Arduino is capable of supplying this potential difference, but the current that these devices draw can often exceed what the Arduino can safely output. Therefore, it is recommended that an external power supply is used to power the servo, such as a power supply module or battery pack.

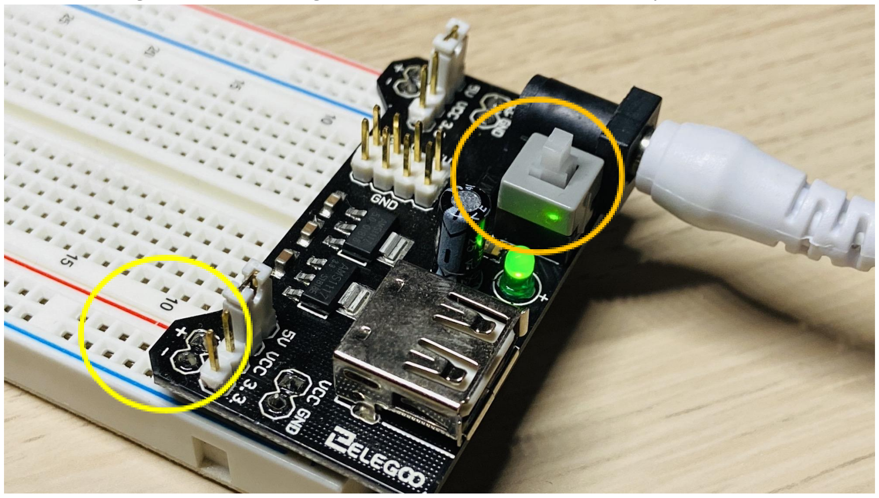
In this course, we will use this symbol to represent a servo motor in schematic (circuit) diagrams. The three lines represent the three wires of the servo. The left one is the power wire (red), the right one is the ground wire (black/brown), and the top one is the signal wire (white/orange).



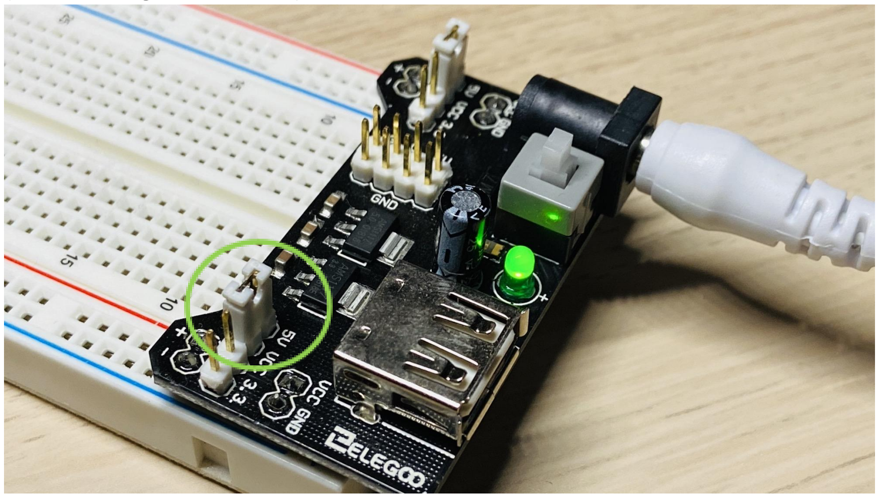
Part 1: Hardware

1. Gather the following materials:
    a. One Arduino, USB Cable, and Breadboard
    b. Four Jumper Wires
    c. Standard servo + arms
    d. 9 V battery and battery wire
    e. Power supply module

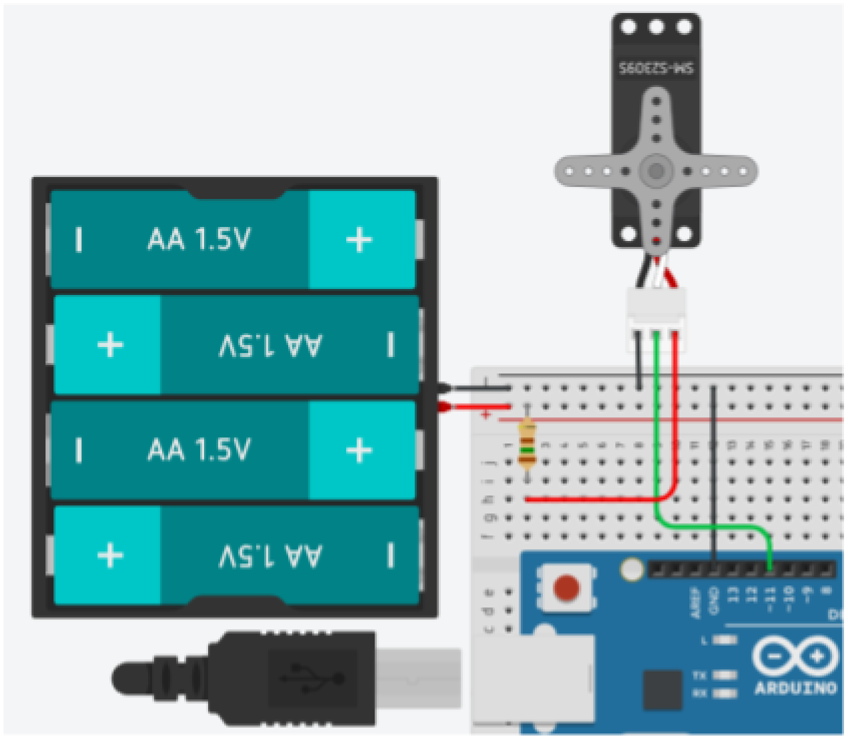
2. Connect one of the arms to the servo so that you can more easily see it rotate
3. Plug the power supply module into the breadboard, as shown below. Make sure to align the positive and negative markings with the correct rails of your breadboard (circled in yellow).



4. Connect a 9V battery to a battery wire and plug it into the power supply. When you press down the button (circled in orange) the green LED should light.
5. Verify that the white jumper is set to 5V rather than 3.3V for the power rail that you plan to use (circled in green below).

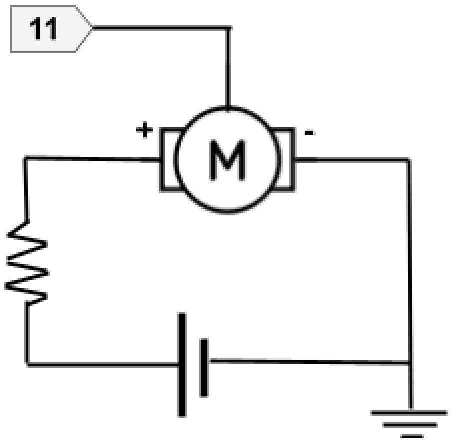


6. Connect the GND pin on the Arduino to the ground rail of the breadboard in order to ensure a common ground.
7. Connect the servo to the system. Connect the left (brown) pin to the (-) column. Connect the middle (red) pin to the (+) column. Connect the right (orange) pin to Digital Pin 11 on the Arduino.



NOTE:
This diagram is for TinkerCad. If you are using a power supply module that outputs 5V, you do not need to include an additional resistor.

Schematic (Circuit) Diagram



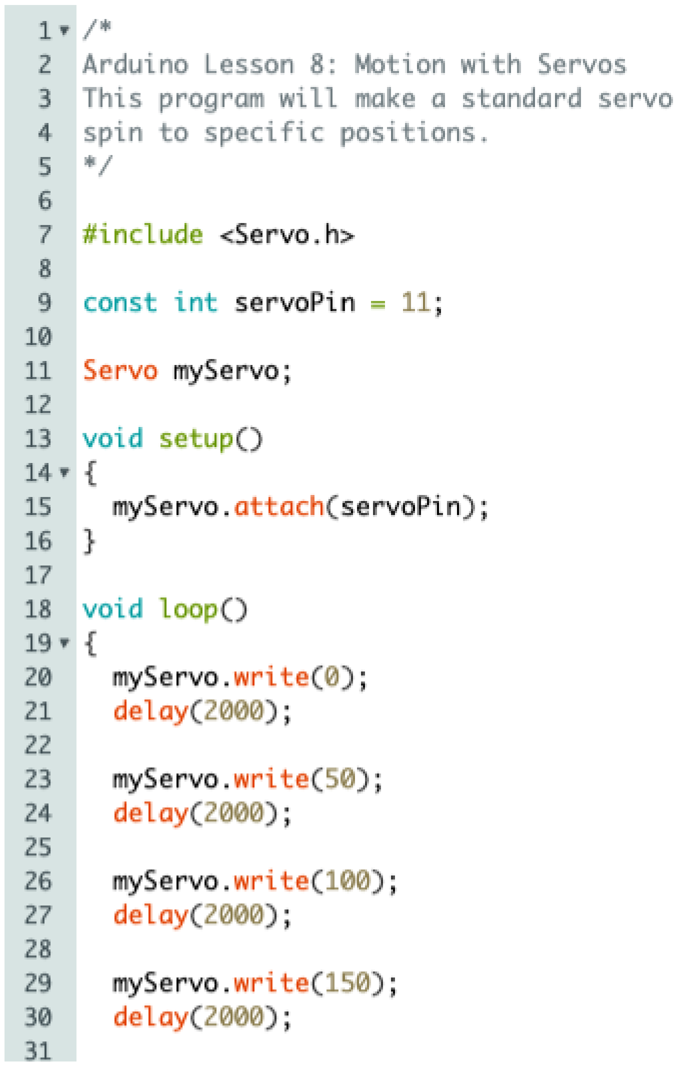
Notice how there is an external power supply (the battery) and the ground for Arduino represented in the diagram. Also, servos are "polar" so orientation matters.

Part 2: Software

8. Copy+Paste or type up the program for this lesson.

```
#include <Servo.h>

const int servoPin = 11;

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

void setup() {
  myservo.attach(servoPin);  // attaches the servo on pin 11 to the servo object
}
```

```
void loop() {
  myservo.write(0);
  delay(2000);

  myservo.write(50);
  delay(2000);

  myservo.write(100);
  delay(2000);

  myservo.write(150);
  delay(2000);

  myservo.write(100);
  delay(2000);

  myservo.write(50);
  delay(2000);
}
```



```
/*
Arduino Lesson 8: Motion with Servos
This program will make a standard servo
spin to specific positions.
*/

#include <Servo.h>

const int servoPin = 11;

Servo myServo;

void setup()
{
  myServo.attach(servoPin);
}

void loop()
{
  myServo.write(0);
  delay(2000);

  myServo.write(50);
  delay(2000);

  myServo.write(100);
  delay(2000);

  myServo.write(150);
  delay(2000);

  myServo.write(100);
  delay(2000);

  myServo.write(50);
  delay(2000);
}
```

9. Save the file**.** Verify your code and troubleshoot any errors that arise. Then upload this program to the Arduino.

10. Wait a few seconds and watch. You should see the servo spin back and forth to specific positions.

Documenting the Original Task

Now take a video of your functioning setup and link it here (upload it to Google Drive first) OR link your TinkerCad simulation here.

Reflection

Let's look at the code. On Line 7 is something new: **#include <Servo.h>** . This line of code tells the Arduino to go get the Servo Library - a collection of coding structures that are designed specifically for servo motors. Without this line of code, the Arduino wouldn't recognize the new functions that are used.

Look at Line 11: **Servo mySservo**. Here we are creating a "Servo object" with the name *myServo*. In a sense, this is similar to declaring a variable ("int x"). **Servo** is kinda like a new data type, and *myServo* is kinda like a variable of this data type. But Servo objects have special properties and unique functions that can be used with them, so *myServo* can do much more than just hold a value. Also, instead of the name *myServo*, you could have used whatever name you wanted (even something as simple as "x").

We are using Pin 11 as an output, but notice we never used the pinMode( ) function. Instead, in Line 15 the function **attach( )** was used. This function takes care of everything needed to get Pin 11 ready to control the servo.

Finally, notice how the new functions are written: they always follow the object *myServo* and a period (eg, *myServo***.***attach( )*). These new functions are methods that belong to these objects (in a sense they are "inside" of the objects). We access functions that belong to objects using this notation. You can never use **attach( )** or **write( )** by themselves, since they only work with/on a Servo object.

Research

1. Take a look at the **Arduino Servo Library**.

   a. Which two pins (on an Arduino Uno board) are affected by using the Servo Library? In what way are they affected?

   b. What does the **write( )** function do? What are the lowest and highest numbers that can be used as arguments?

2. We are using a *standard* servo, but there is another popular kind called **continuous**. Look it up. How are continuous servos different from standard servos? (Write at least three sentences.)

Trouble-Shooting

Be sure to correct each error before moving on.

Error To Make	**Message & Reasoning**
Delete Line 7	

Part 3: Challenges (Pick Two)

Complete 2+ challenges. Remember that for each you must include a video of your functioning design with your names in it, and copy+paste your code.

1. Use a *for* loop to make the servo sweep back and forth continuously rather than in noticeable steps.

   Paste your code here:

Take a video of your circuit. Include your name. Paste it here or upload it to this Google Classroom assignment.

2. Dance! Create a “dance” routine for the servo (you can check out this video for some inspiration).

   Paste your code here:

Take a video of your circuit. Include your name. Paste it here or upload it to this Google Classroom assignment.

3. Add two LEDs to the original system. Add to the code such that when the servo gets to one extreme (0^o), the first LED should go on. When the servo gets to the other maximum (180^o), the second LED should go on. Draw a schematic (circuit) diagram to go with this.

   Paste your code here:

Take a video of your circuit. Include your name. Paste it here or upload it to this Google Classroom assignment.

Schematic/circuit diagram:

4. Add a potentiometer to the system. Modify the code such that the potentiometer controls the position of the servo. For example, when the potentiometer is turned to one extreme (all the way left), the servo position should be at 0º. When the potentiometer is turned to the other extreme (all the way right), the servo position should be at 180º. (You'll want to create a function to convert the reading from the potentiometer into a value that can be used by the servo.) Draw a schematic (circuit) diagram to go with this.

   Paste your code here:

Take a video of your circuit. Include your name. Paste it here or upload it to this Google Classroom assignment.

Schematic/circuit diagram:

5. Add a photoresistor to the system. Modify the code such that the servo motor stays in one location (0º) while the photoresistor is uncovered. When the photoresistor is covered (you choose the threshold), the servo should rotate to another location (90º). Draw a schematic (circuit) diagram to go with this.

   Paste your code here:

Take a video of your circuit. Include your name. Paste it here or upload it to this Google Classroom assignment.

Schematic/circuit diagram:

6. Create three different functions. Each function should contain a unique “routine” for the servo (like a dance). Add a potentiometer, which will control which routine is currently “playing”. When the potentiometer is between 0 and 300, the first routine should be executed. When the potentiometer is between 300 and 600, the second routine should be executed. And when the potentiometer is between 600 and 1023, the last routine should be executed.

   Paste your code here:

Take a video of your circuit. Include your name. Paste it here or upload it to this Google Classroom assignment.

Schematic/circuit diagram:

Additional Resources

To learn more about servos, you can check out these resources:

- **Tutorial 05 for Arduino: Motors and Transistors** by Jeremy Blum (jump to 7:30)
- **Arduino Cookbook** by Michael Margolis (Chapter 8: Physical Output)
- **How Servo Motors Work** by How To Mechatronics

Arduino Servo Motors KEY

1. Take a look at the **Arduino Servo Library**.
    a. Which two pins (on an Arduino Uno board) are affected by using the Servo Library? In what way are they affected?

       Answer: pins 9 and 10; when using the Servo library you can't use pulse-width modulation for those pins (i.e. you can't use analogWrite() to have a voltage other than 0V/5V.

    b. What does the **write( )** function do? What are the lowest and highest numbers that can be used as arguments?

       Answer: For a normal servo this sets the angle. For a continuous servo this sets the speed. It takes numbers from 0 to 180.

2. We are using a *standard* servo, but there is another popular kind called **continuous**. Look it up. How are continuous servos different from standard servos? (Write at least three sentences.)

    Answer: a standard servo only spins 180 degrees while a continuous servo can spin indefinitely.

Trouble-Shooting

Be sure to correct each error before moving on.

Error To Make	Message & Reasoning
Delete Line 7	*Answer: 'Servo does not name a type; did you mean 'Serial'?'* *This happens because without incorporating the servo library, Servo isn't defined. The computer doesn't know what it means.*