How Servo Motor Works & Interface With Arduino

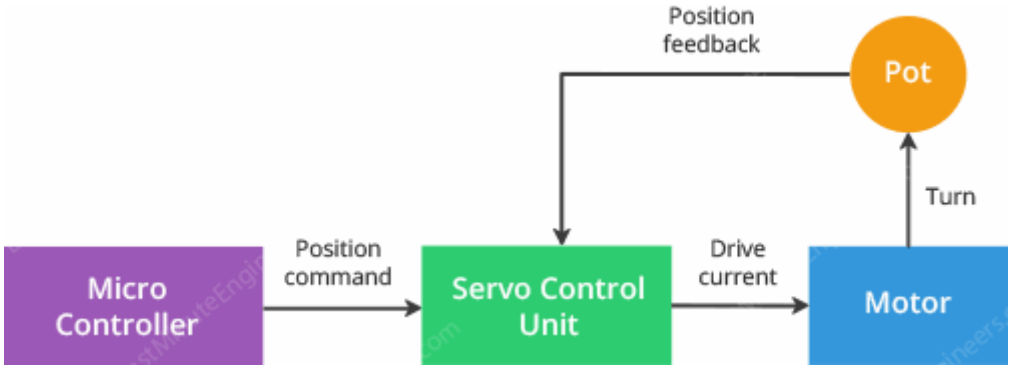
www. https://lastminuteengineers.com/servo-motor-arduino-tutorial/

Want to add motion to your next Arduino project without building a motor controller? Then servo motors might be the solid launching point for you.
Unlike DC motors, you can precisely control the positioning of these motors. Instruct them where to point, and they'll do it for you.

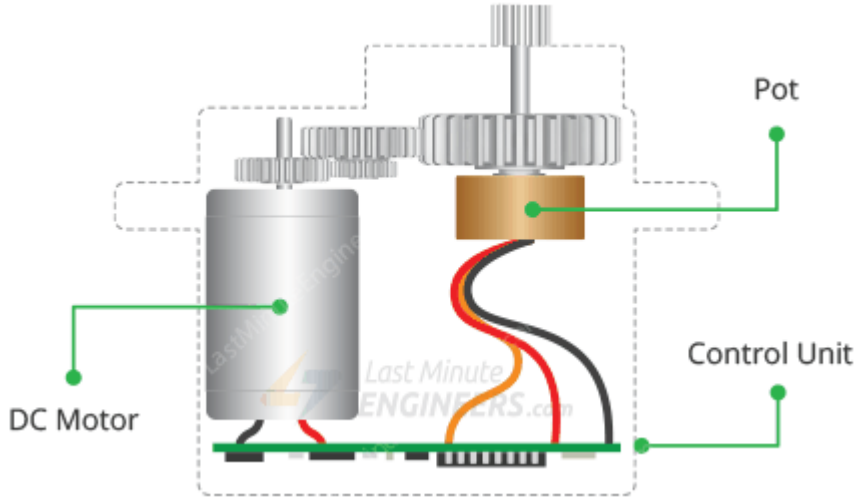
They're useful in many robotics projects, such as for turning the front wheels on an RC model for steering or pivoting a sensor to look around on a robotic vehicle.

What is Servo?

Servo is a general term for a closed loop control system.
A closed loop system uses the feedback signal to adjust the speed and direction of the motor to achieve the desired result.



RC servo motor works on the same principal. It contains a small DC motor connected to the output shaft through the gears.
The output shaft drives a servo arm and is also connected to a potentiometer (pot).

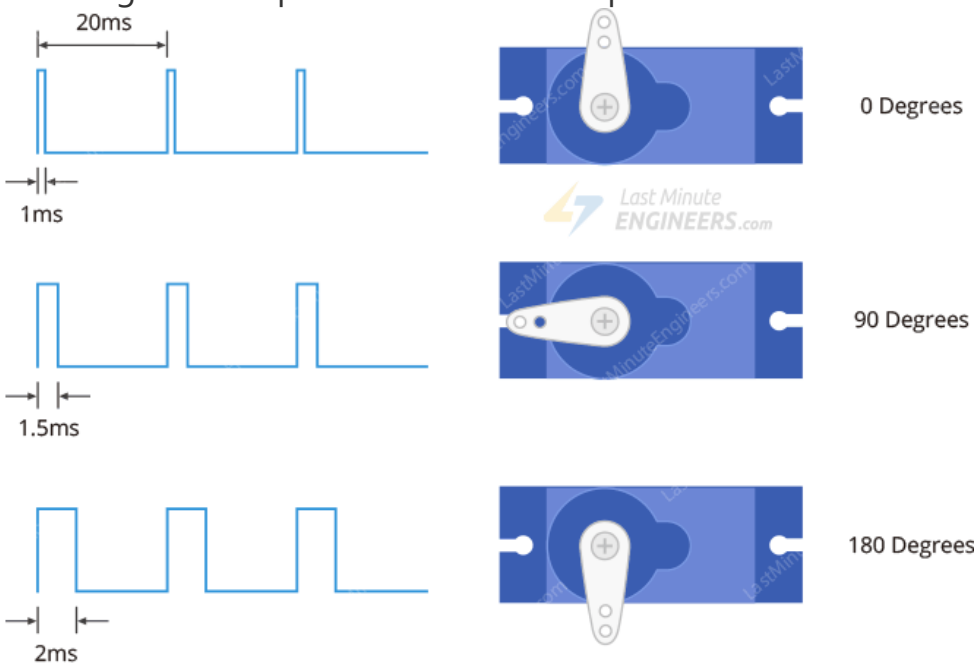


The potentiometer provides position feedback to the servo control unit where the current position of the motor is compared to the target position.

According to the error, the control unit corrects the actual position of the motor so that it matches the target position.

How Servo Motors Work?

You can control the servo motor by sending a series of pulses to the signal line. A conventional analog servo motor expects to receive a pulse roughly every 20 milliseconds (i.e. signal should be 50Hz).
The length of the pulse determines the position of the servo motor.

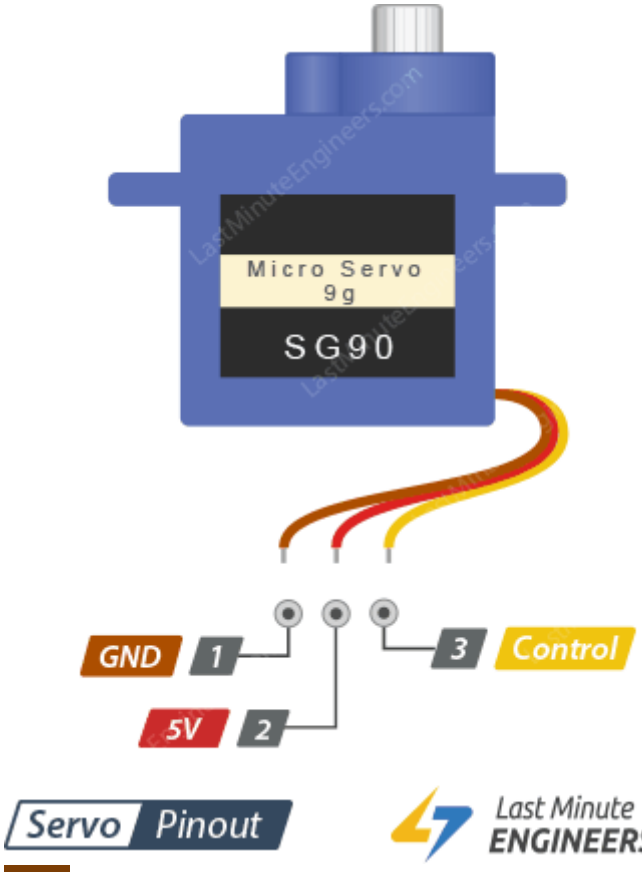


- If the pulse is high for 1ms, then the servo angle will be zero.
- If the pulse is high for 1.5ms, then the servo will be at its center position.
- If the pulse is high for 2ms, then the servo will at 180 degrees.
- Pulses ranging between 1ms and 2ms will move the servo shaft through the full 180 degrees of its travel.

The duration of the pulses may sometimes vary with different brands and they can be 0.5ms for 0 degrees and 2.5ms for 180 degrees.

Servo Motor Pinout

Servo motors typically have three connections and are as follows:

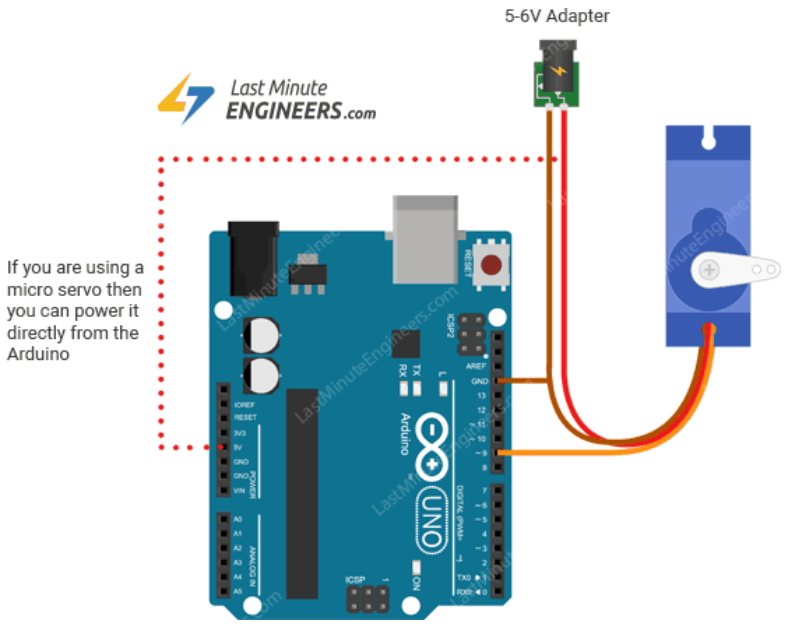


GND is a common ground for both the motor and logic.
5V is a positive voltage that powers the servo.
Control is input for the control system.

The color of the wires varies between servo motors, but the red wire is always 5V and GND will either be black or brown. The control wire is usually orange or yellow.

Wiring Servo Motor to Arduino UNO

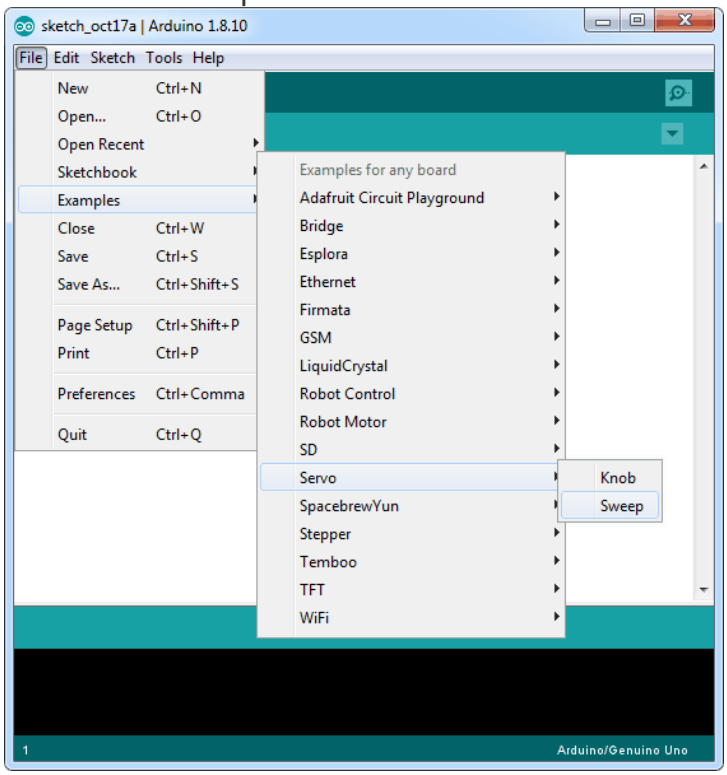
Let's hook the servo motor up to the Arduino.
For example let's use SG90 Micro Servo Motor. It runs on 4.8-6VDC (5V Typical) and can rotate approximately 180 degrees (90 in each direction).
It consumes around 10mA at idle and 100mA to 250mA when moving, so we can power it up through 5-volt output on the Arduino.
If you have a servo that consumes more than 250mA, consider using a separate power supply for your servo.
Connect the Red wire to the 5V on Arduino (or DC jack) and Black/Brown wire to ground.
Finally connect the Orange/Yellow wire to the PWM enabled pin 9.



Arduino Code – Sweep

For our first Arduino sketch, we will use one of the built-in examples that come with the Arduino IDE.

Go to the Examples sub-menu. Select the Servo and Load the Sweep sketch.



Go ahead and upload the sketch. You will immediately see the motor moving in one direction and then going back in another.

```
#include <Servo.h>

int servoPin = 9;
Servo servo;
int angle = 0;  // servo position in degrees

void setup() {
    servo.attach(servoPin);
```

```
}

void loop() {

    // scan from 0 to 180 degrees
    for(angle = 0; angle < 180; angle++) {
        servo.write(angle);
        delay(15);
    }

    // now scan back from 180 to 0 degrees
    for(angle = 180; angle > 0; angle--) {
        servo.write(angle);
        delay(15);
    }
}
```

Explanation:

Controlling servos is not an easy task, but luckily for us, Arduino IDE already contains a very nice library called Servo. It includes simple commands so that you can quickly instruct the servo to turn to a particular angle.

If you are going to use these commands, you need to tell the Arduino IDE that you are using the library with this command:

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

The next thing we do is declare the Arduino pin to which the control pin of the servo motor is connected.

```
int servoPin = 9;
```

Below line creates a servo object.

```
Servo servo;
```

You can actually define up to eight servos in this way, for example, if we had two servos, then we could write something like this:

```
Servo servo1;
Servo servo2;
```

The variable `angle` is used to store the current angle of the servo in degrees.

```
int angle = 0;
```

In the setup function, we link the `servo` object to the pin that will control the servo using this command:

```
servo.attach(servoPin);
```

The loop function actually contains two for loops. The first loop increases the angle in one direction and the second in the opposite direction.

Below command tells the servo to update its position to the specified angle.

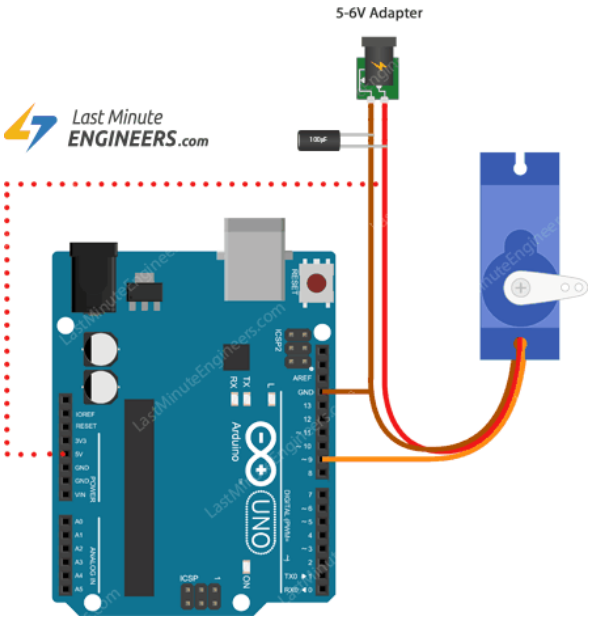
```
servo.write(angle);
```

Troubleshooting

Sometimes your servo may misbehave if you decide to run it directly from the Arduino. The reason for this is that the servo draws considerable power, especially during start-up, and this can cause the Arduino board to reset.

If this happens, you can usually fix this by placing a fairly large electrolytic capacitor (470uF – 1000uF) between GND and 5V.

The capacitor acts as a reservoir of electricity, so that when the motor starts, it takes charge from the capacitor as well as the Arduino supply.
The longer lead of the capacitor should be connected to 5V and the negative lead to GND.

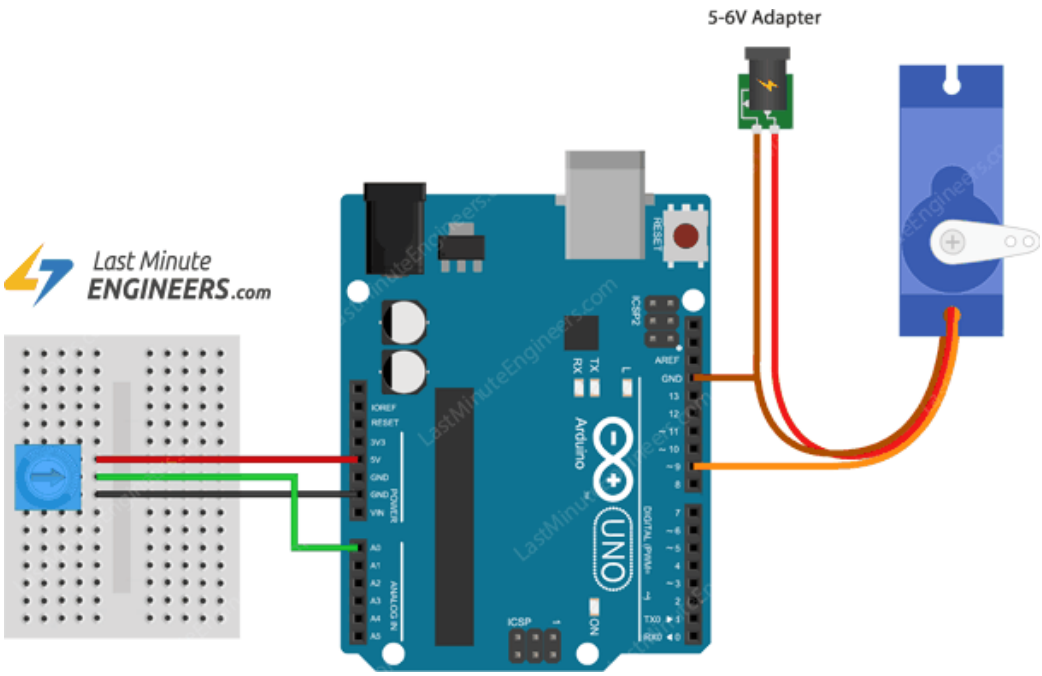


Controlling Servo with a Potentiometer

Our next step is to add a potentiometer so that we can control the position of the servo by turning the knob.
This project can be very useful when you want to control the pan and tilt of a sensor connected to the servo.

Wiring

As the wiring diagram shows you'll need a potentiometer, any value from 10k up will be OK. Connect one end of the pot to ground, the other end to the Arduino 5V and the wiper to analog input A0.



Arduino Code

The code to make the servo follow the knob's position is simpler than to make it sweep.

```
#include <Servo.h>

int potPin = 0;
int servoPin = 9;
Servo servo;

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

void loop() {
    int reading = analogRead(potPin);
    int angle = map(reading, 0, 1023, 0, 180);
    servo.write(angle);
}
```

Notice that there is now a new variable called `potPin`.

In the loop function, we start by reading the value from the analog pin A0.

```
int reading = analogRead(potPin);
```

This gives us a value of between 0 and 1023. But we need to scale it down, since the servo can only rotate through 180 degrees.

One way to do this is to use the Arduino map() Function which re-maps a number from one range to another. So, below line changes the reading to represent the angle between 0 and 180 degrees.

```
int angle = map(reading, 0, 1023, 0, 180);
```

Finally, we use the `write()` command that tells the servo to update its position to the angle selected by the potentiometer.

```
servo.write(angle);
```