Arduino Servo Motors

by **cornelam** on July 2, 2015

Table of Contents

Author:cornelam
Just somebody who believes DIY is better. Oh, and check my new book, Arduino Development Cookbook here: goo.gl/ANmHRe

Intro: Arduino Servo Motors

Servo motors are great devices that can turn to a specified position.

Usually, they have a servo arm that can turn 180 degrees. Using the Arduino, we can tell a servo to go to a specified position and it will go there. As simple as that!

Servo motors were first used in the Remote Control (RC) world, usually to control the steering of RC cars or the flaps on a RC plane. With time, they found their uses in robotics, automation, and of course, the Arduino world.

Here we will see how to connect a servo motor and then how to turn it to different positions.

The first motor I ever connected to an Arduino, seven years ago, was a Servo motor. Nostalgic moment over, back to work!

We will need the following things:

1. An Arduino board connected to a computer via USB
2. A servo motor
3. Jumper wires

There are few big names in the servo motor world. Hitec and Futaba are the leading RC servo manufacturers. Good places to buy them are Servocity, Sparkfun, and Hobbyking.

This instructable and many more can be found in my Arduino Development Cookbook available here. :D

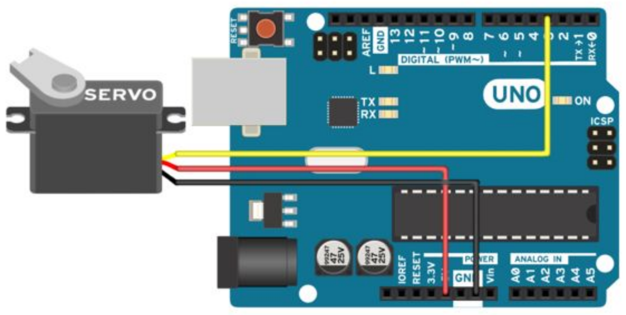
Step 1: How to connect them

A servo motor has everything built in: a motor, a feedback circuit, and most important, a motor driver. It just needs one power line, one ground, and one control pin.

Following are the steps to connect a servo motor to the Arduino:

1. The servo motor has a female connector with three pins. The darkest or even black one is usually the ground. Connect this to the Arduino GND.
2. Connect the power cable that in all standards should be red to 5V on the Arduino.
3. Connect the remaining line on the servo connector to a digital pin on the Arduino.

Check the image for a view of the servo connected to the Arduino.



Step 2: Code

The following code will turn a servo motor to 0 degrees, wait 1 second, then turn it to 90, wait one more second, turn it to 180, and then go back.

```
// Include the Servo library
#include <Servo.h>
// Declare the Servo pin
int servoPin = 3;
// Create a servo object
Servo Servo1;
void setup() {
  // We need to attach the servo to the used pin number
  Servo1.attach(servoPin);
}
void loop(){
  // Make servo go to 0 degrees
  Servo1.write(0);
  delay(1000);
  // Make servo go to 90 degrees
  Servo1.write(90);
  delay(1000);
  // Make servo go to 180 degrees
  Servo1.write(180);
  delay(1000);
}
```

If the servo motor is connected on another digital pin, simply change the value of servoPin to the value of the digital pin that has been used.

File Downloads

Servo_motor.ino (548 bytes)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Servo_motor.ino']

Step 3: How it works

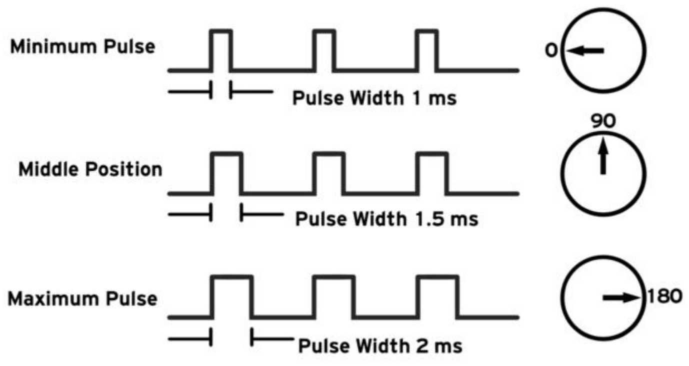
Servos are clever devices. Using just one input pin, they receive the position from the Arduino and they go there. Internally, they have a motor driver and a feedback circuit that makes sure that the servo arm reaches the desired position. But what kind of signal do they receive on the input pin?

It is a square wave similar to PWM. Each cycle in the signal lasts for 20 milliseconds and for most of the time, the value is LOW. At the beginning of each cycle, the signal is HIGH for a time between 1 and 2 milliseconds. At 1 millisecond it represents 0 degrees and at 2 milliseconds it represents 180 degrees. In between, it represents the value from 0–180. This is a very good and reliable method. The graphic makes it a little easier to understand.

Remember that using the Servo library automatically disables PWM functionality on PWM pins 9 and 10 on the Arduino UNO and similar boards.

Code breakdown

The code simply declares the servo object and then initializes the servo by using the servo.attach() function. We shouldn't forget to include the servo library. In the loop(), we set the servo to 0 degrees, wait, then set it to 90, and later to 180 degrees.



Step 4: More things about servos

Controlling servos is easy, and here are a few more tricks we can use:

Controlling the exact pulse time

Arduino has a built-in function servo.write(degrees) that simplifies the control of servos. However, not all servos respect the same timings for all positions. Usually, 1 millisecond means 0 degrees, 1.5 milliseconds mean 90 degrees, and, of course, 2 milliseconds mean 180 degrees. Some servos have smaller or larger ranges.

For better control, we can use the servo.writeMicroseconds(us) function, which takes the exact number of microseconds as a parameter. Remember, 1 millisecond equals 1,000 microseconds.

More servos

In order to use more than one servo, we need to declare multiple servo objects, attach different pins to each one, and address each servo individually. First, we need to declare the servo objects—as many as we need:

```
// Create servo objects
Servo Servo1, Servo2, Servo3;
```

Then we need to attach each object to one servo motor. Remember, every servo motor uses an individual pin:

```
Servo1.attach(servoPin1);
Servo2.attach(servoPin2);
Servo3.attach(servoPin3);
```

In the end, we just have to address each servo object individually:

```
Servo1.write(0); // Set Servo 1 to 0 degrees
Servo2.write(90); // Set Servo 2 to 90 degrees
```

Connection-wise, the grounds from the servos go to GND on the Arduino, the servo power to 5V or VIN (depending on the power input), and in the end, each signal line has to be connected to a different digital pin. Contrary to popular belief, servos don't need to be controlled by PWM pins—any digital pin will work.

Continuous rotation servos

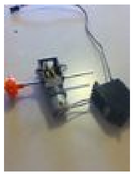
There is a special breed of servos labelled as **continuous rotation servos**. While a normal servo goes to a specific position depending on the input signal, a continuous rotation servo either rotates clockwise or counter-clockwise at a speed proportional to the signal. For example, the Servo1.write(0) function will make the servomotor spin counter-clockwise at full speed. The Servo1.write(90) function will stop the motor and Servo1.write(180) will turn the motor clockwise at full speed.

There are multiple uses for such servos; however, they are really slow. If you are building a microwave and need a motor to turn the food, this is your choice. But be careful, microwaves are dangerous!

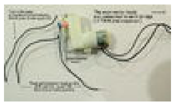
Step 5: Check out more

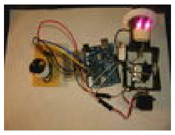
More topics regarding motors such as brushless, transistor drivers ormotor speed control can be found in my Arduino Development Cookbook available here. :D

Related Instructables

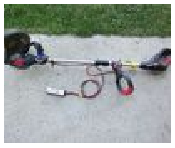

Jury Rigging A Servo To Drive A Low Voltage Motor (DIY ESC) by Nicapizza

ROBOTIC ARM Arduino Controlled by Evil zoids


Make your own low-cost servo by electro18


very basic pan tilt using arduino by katman101

Robotic Spider V8 by FoamboardRC


Weed trimmer motor replaced with brushless RC truck motor and uses an Arduino control board by AnEngineersView

Comments