04 - Servos 'n More

In this project, we are going to introduce the use of a special type of motor called a Servo. These motors are great for precision movement of things like latches, grippers, or other linkages.

SKILL REQUIREMENTS

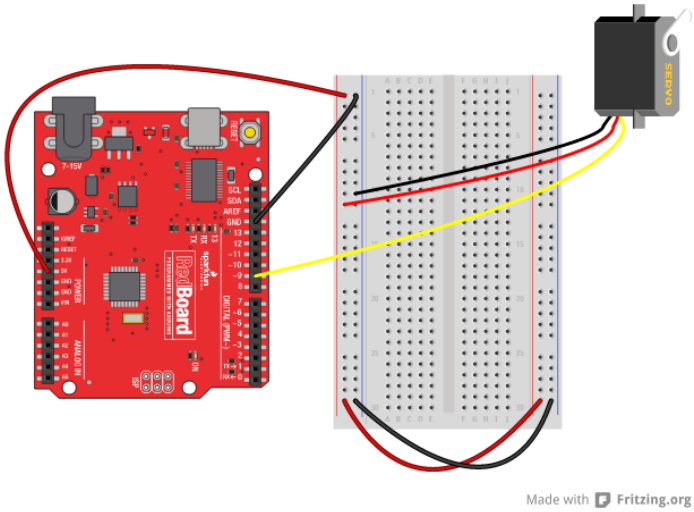
- ELECTRICAL PROTOTYPING
- ROBOTICS
- SOLDERING
- PROGRAMMING
- DIY

MATERIALS LIST

- Computer
- Arduino UNO
- USB Cable
- Servo
- Breadboard

STEP 1: Wiring / Background

The wiring for this circuit is pretty simple. The Servo motor uses just three wires. One for Power (5V), One for Ground (GND), and the third for a Signal.



Before uploading any code to your board, make sure you attach a servo horn onto the end of the servo motor. Servo horns come in many different shapes / sizes. Pick one that you can easily rotate and see. After you attach the servo horn, gently turn the horn. The servo has a limited range of motion. Roughly how far does it rotate?

STEP 2: Example Code

Open up this example code (codebender.cc/sketch:136780), and upload it to your board. After the upload is complete, you should notice that the servo motor jumps immediately to a position. Now, let's look at the code. Here is the code (without comments)

```
#include <Servo.h>

Servo myServo;

void setup()
{
    myServo.attach(9);
}

void loop()
{
    myServo.write(90);
}
```

The first line includes the `Servo.h` library into the sketch. This provides access to new commands and functions to control special devices - like a servo motor. Notice that this is one of the few instances where there is no semicolon at the end of the line.

Next, **`Servo myServo`** - this creates an ***object*** that is called myServo. An object is similar to a variable that has extra properties and commands. The two commands that we care about are: **`.attach()`** and **`.write()`**.

The **`myServo.attach(9)`** is similar to a setup command. It links the `myServo` object to pin 9 on the Arduino.

Finally, **`myServo.write(90)`** is the command that tells the servo motor what angle to rotate to. The range of motion for a servo is roughly 180 degrees. You can use any value from 0 to 180 with this command.

Set the angle to either 0 or 180 and click upload. Do you feel the motor chattering / shaking? This is because the motor is hitting a hard stop and is stalled out. Be careful with this. Avoid pushing the motors to these limits. It *can* damage the motors. Try a few other angles until you find a good limit for your motor. Write these down here:

Minimum Angle: _____ Maximum Angle: _____

STEP 3: Servo Blink

Now, modify your code so that the Servo motor "blinks" or moves from the minimum angle to the maximum angle. Make sure that you have a delay that's long enough for the motor to actually rotate to that position.

If you get stuck, we have a hint here: codebender.cc/sketch:136778

Challenge Project:

- Connect up a puppet / doll / character to the servo motor horn. Program your character to dance and move to a song!

STEP 4: Serial Servo Control

Often we want to control things without having to re-upload code. This example will show you how you can use the Serial communication interface to control the motor angle. This example has all of the comments removed - for the full code example go to: codebender.cc/sketch:73819

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

void setup()
{
  myServo.attach(9);
  Serial.begin(9600);
  Serial.println("Type in an angle.");
}

void loop()
{
  int angle;
  if(Serial.available() > 0)
  {
    angle = Serial.parseInt();
    myServo.write(angle);
    Serial.print("Setting angle to: ");
    Serial.println(angle);
  }
}
```

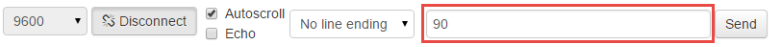
Most of the code is the same as our example above. We setup the servo motor on pin 9, open up the Serial communication at 9600 bits/sec, and print out a short prompt for the user.

In the loop, we have an `if()` statement that checks to see if data is "available" on the Serial interface. This is data that you will send from your computer.

If data is available, we read in an integer value using the `Serial.parseInt()` command and use this value to set the angle of the Servo.

Finally, for feedback to the user, the program prints out to the Serial monitor what it just did.

Let's test it out. Upload this code, and open up the Serial Monitor. Type in the angle and click [Send]. Make sure that you change the option to "**No line ending**"



This piece of code is a handy tool to have so that you don't have to re-compile the code each time. It's also useful when setting up and calibrating servos for use with a project.

Going Further -- Adding More Servos

The Servo library provides a "blueprint" for a Servo motor and allows us to create multiple objects that use this same blueprint. To add a second servo, connect up a second servo using the previous diagram as a guide. Connect the white wire (signal) to a different pin - such as pin 10 on the Arduino. Now, add this line near the top of your code right after the `Servo myServo;`

```
Servo myServo2;
```

In the `setup()`, initialize myServo2 using the command: `myServo2.attach(10);` Make sure that the pin number corresponds with your wiring. Set the angle for this servo, using `myServo2.write(angle);`