Servo motor control © 2023 by Ivan Novosel is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

ROBOTICS

PHYSICAL COMPUTING

Servo motor control

Author: Ivan Novosel

Institution: V. gymnasium, Zagreb

Co-funded by the Erasmus+ Programme of the European Union

Disclaimer: This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Introduction

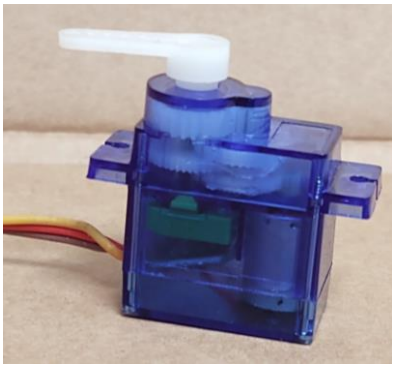
This is the first time we will be using motors of any kind. We will start with the ones which are easiest to use – servo motors. We will build a few introductory circuits using servo motors, and learn a few important things about using them effectively:

1. Sweep – how to sweep from one side to another.
2. Servo + potentiometer – how to tell the servo where to turn using a simple sensor.
3. Calibrating the servo motor

Servo motors

If we want to have any kind of robot that has moveable parts, we need motors. What kind we choose depends on the use.

Servo motor is a motor combined with some means of control of position – it has a sensor for positional feedback.



Picture 1 – What is inside of the servo motor: on top is the horn, it is attached to the gears inside the motor. Just below the gears is the sensor for positional feedback. The actual motor is to the right of it. The gears transfer the rotation of the motor to the horn.

For that reason, their main use is for situations when we must position a component precisely. They have a limited range of motion, usually from 0 to 180 degrees, but it depends on the particular motor.

At our disposal we have small hobby servo motors which have very limited power and precision. Even with their limited need for power they easily take more than Arduino can handle so they can behave erratically.

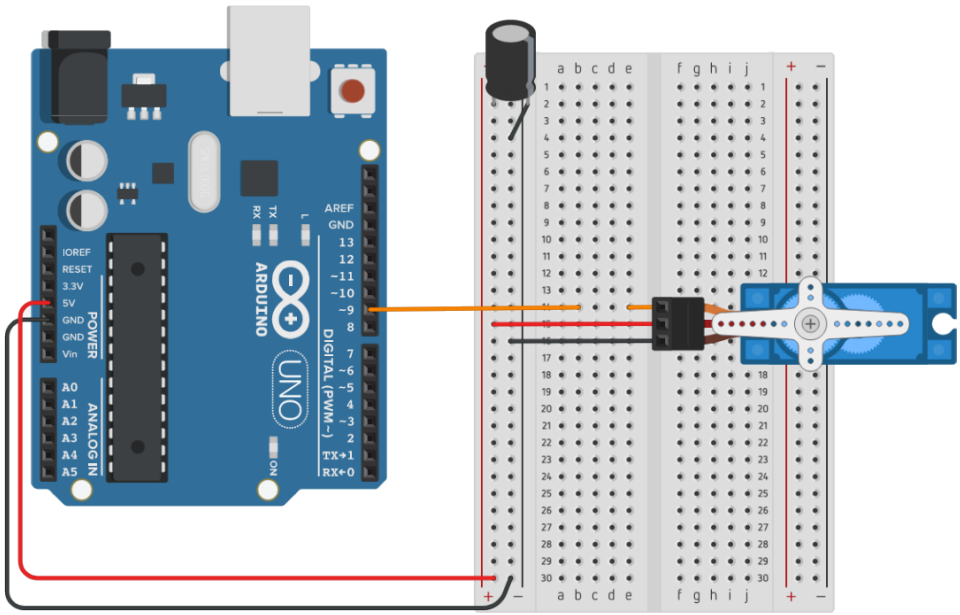
Don't be alarmed if your motor behaves erratically!

Hobby servo motors are controlled with PWM signal. It moves depending on how long the signal is and stays at a certain position (angle). We will not be timing the servos manually – it is much simpler to use the Servo library that comes preinstalled with Arduino IDE.

Moving the servo motor

First, we need to connect the servo motor. All hobby servo motors have 3 wires:

- red wire – power, connect to + side of power source.
- brown or black – ground, connect to ground of the source.
- orange or yellow – signal, connect to PWM enabled pin.



Picture 2 – wiring up a SG90 micro servo to Arduino. We are going to connect additional components later but for now we are connecting just a capacitor and the servo motor.

When you connect all the components try it out with programming a basic example:

```
#include <Servo.h>

Servo myservo;

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

void loop() {
  myservo.write(0);
  delay(100);
  myservo.write(90);
  delay(100);
  myservo.write(180);
  delay(100);
}
```

TASK: Run the example and read the code!

This should move the servo to 3 different angles in a loop.

Sweep the servo motor

Simulation of **Sweep example** on Tinkercad, URL: https://bit.ly/3smuFPo

Now we will see how we can move our servo motor more continuously. For this we will study one of the examples that come with Arduino IDE, you can open it through *File → Examples → Servo → Sweep*

We don't need to connect anything new for this example, we will use the same components, but we will see how we can use a for loop to change the angle of the servo.

TASK: Run the example and read the code!

The loop will increase and decrease the angle of the servomotor gradually.

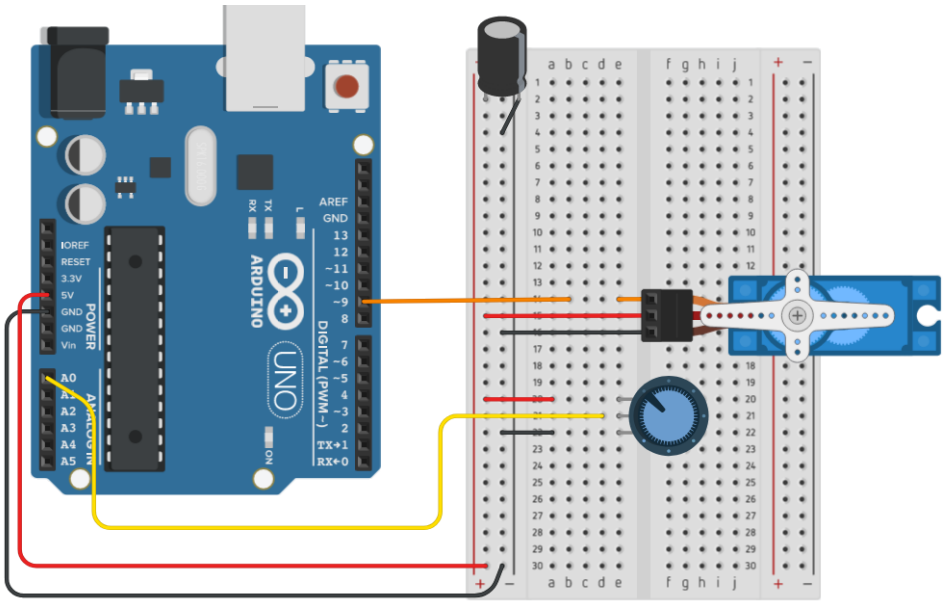
HINT: run the simulation to see how the PW signal changes on the oscilloscope.

Servo motor with potentiometer

Potentiometer is a variable resistor, it changes its value depending on how you turn it.

Simulation of **Knob example** on Tinkercad, URL: https://bit.ly/3QNTRHO

We have seen how to control the servo motor through direct commands, now we will see a version where we use a simple sensor to control it. Potentiometer is used as a simple rotary sensor, and we use the signal from it, convert it into angles, and drive the servo.



Picture 3 – wiring up a SG90 micro servo with a potentiometer. Potentiometers always have 3 pins, so even if it looks differently the middle pin is the one that you want to connect to A0 pin of Arduino. The rest is like in the picture 1.

To make this work we will use the other example from Arduino IDE, open the *File → Examples → Servo → Knob*

Calibrating servo motors

Calibration – process of comparing the measures given by the device we are testing to a reference device that has correct measures.

If we want to use servo motors to rotate things precisely ideally, we would use an encoder – encoder is an additional sensor that measures the angle of the servo motor. But even with a hobby servo motor **we can get more precision out of it if we calibrate it.**

For this we need an external measure of angle – use the 3D printed protractor and pointer.

Picture 4 – use the screws to fix the protractor and pointer to the servo motor as it is show in the picture. **Watch out that you turn the servo in the right direction**, so that the shaft of the servo is in the center of protractor!

Use the potentiometer! DO NOT move the horn of the servo motor while it is powered! It can damage the servo and the Arduino!

We will use the Knob example to help us move the servo. Steps are:

1. Move to 0° - if the pointer is noticeably off the 0° mark unscrew the pointer and reinsert the horn so it is closer to 0°. You probably inserted the horn wrong the first time, this is very common.
2. Move to 180° - for now, just note if it is pointing more or less than 180° if it is off.
3. Change the `attach()` function - Arduino is controlling the servo by sending it differently timed PWM pulses. The default for 0° is 544 (minimum), and for 180° it is 2400 (maximum). We can fine tune the position of 0 and 180 by changing these numbers. We can set this by adapting the `attach()` function so it looks like this: `myservo.attach(pin,min,max)`.
   For example: `myservo.attach(9,544,2450);` would move the position of 180° further back if it didn't reach it.
4. Repeat step 3 until you are happy with the positions.

If you change the min and max in `attach()` you need to upload the code before it takes

Possible problems with servos

If the servo is jittery, it is possible that you have a bad connection between the potentiometer and Arduino, or that the potentiometer isn't properly plugged in the breadboard. Arduino is then reading erratic input values and sending that to the servo motor.

Jitter can also be caused by our programming. If the program blocks for some reason it will change the PWM signal to the motors, which can cause jitter and general erratic movement. To really fix this problem it is good to have a separate controller for servo motors – this is a piece of hardware that generates PWM on its own, and you can usually use more servo motors with it.

You can power small SG90 servos directly from Arduino but **if you use bigger servos or more of them, they will need more power.** The capacitor on the breadboard can help a bit, it is there to smooth out the power spikes when the servo motor starts moving. If you see the power LED on the Arduino blinking that is a sign that motors are drawing too much power. When it loses power Arduino will reset and the program will behave unexpectedly. Sometimes we can fix this by placing a bigger capacitor, but generally it's better that we power the motors separately from the Arduino board.

Capacitor stores electric energy. If a motor draws too much of it from Arduino, the capacitor will behave like an emergency battery.

- **Servo motors can turn to a specific angle and hold it.**
- **Usual range of motion is 180°**
- **Calibrate it to be more precise. The step 1 is the most important!**
- **DO NOT turn the motor by hand while it is powered by Arduino – it will cause damage!**