Jose Rizal University

College of Computer Studies Engineering

Computer Engineering Department

# Simulating Servo Motor Control with Arduino in TinkerCad

**CPE C312 – EMBEDDED SYSTEMS**

**Submitted by:**

Exiquiel John A. Pines

**Submitted to:**

Engr. Nastaran Rezanazarzadeh

**Date Submitted:**

November 9, 2023

**Introduction:**

The purpose of this lab activity is to explore the practical applications of servo motors and their interaction with an Arduino microcontroller. Servo motors are widely used in various projects, including robotics, automation, and model control systems. In this lab, we will perform two distinct activities:

1. Simulating Servo Motor from 0 to 180 Degrees and Back: We will create a circuit and write Arduino code to control a servo motor's position, making it move from 0 to 180 degrees and back. This activity serves as an introduction to servo motor control.

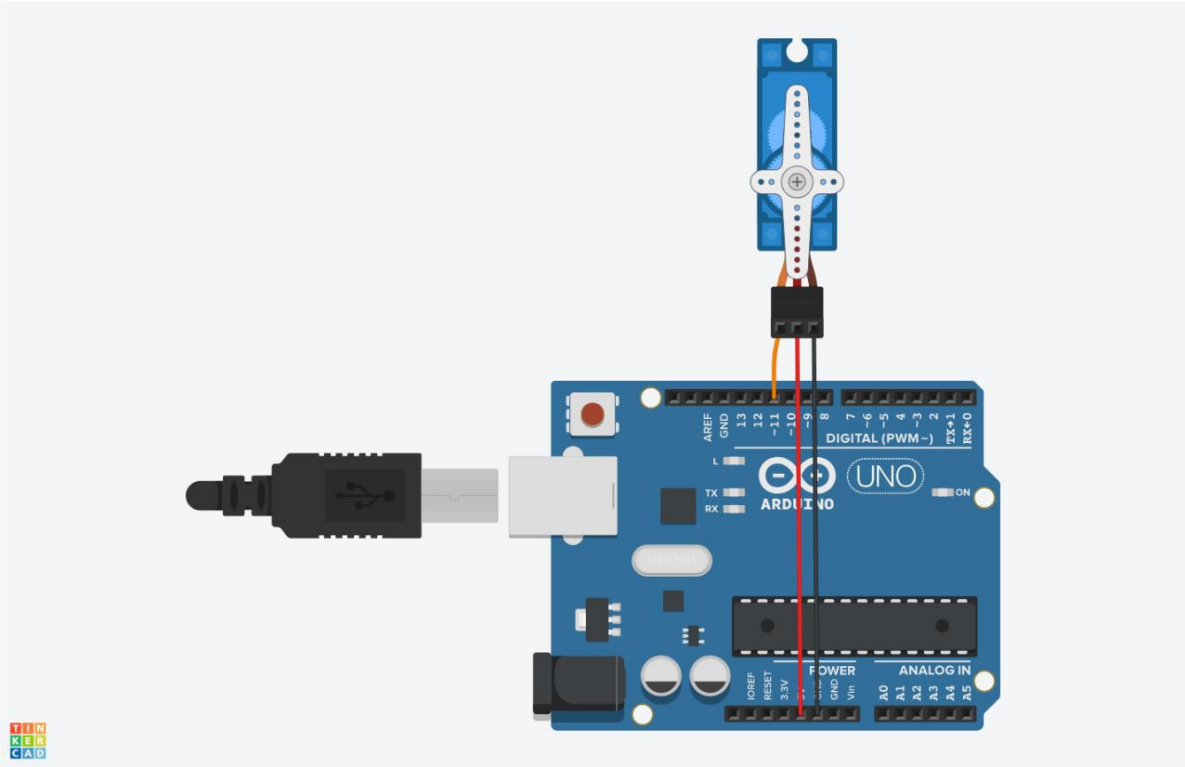
2. Controlling Servo with a Potentiometer: We will enhance the previous circuit by adding a potentiometer, which allows us to control the servo motor's position manually. This activity demonstrates how analog inputs, such as a potentiometer, can be used to adjust servo motor positions dynamically.

## Activity 1: Simulating Servo Motor from 0 to 180 Degrees and Back

### Expected Behavior

In this activity, we expect the servo motor to move smoothly from its initial position (0 degrees) to 180 degrees and then return to 0 degrees. The servo's movement should be precise and controlled, thanks to the Arduino code.

### Circuit Diagram:




### Arduino Code:

```
// Exiquiel John A. Pines
// Simulating Servo Motor from 0 to 180 Degrees and Back

#include <Servo.h>

Servo myservo;

int angle = 0;

void setup()
{
  myservo.attach(11);
}

void loop()
```

```
{
  for (angle = 0; angle <=180; angle +=1) {
    myservo.write(angle);
    delay(10);
  }

  delay(1000);

  for (angle = 180; angle >=0; angle -=1) {
    myservo.write(angle);
    delay(10);
  }

  delay(1000);
}
```

**Observations and Challenges**

- Servo Movement: The servo motor moved precisely as expected, rotating from 0 to 180 degrees and back. The movement was smooth and well-controlled.
- Arduino Code: Writing the Arduino code was relatively straightforward. The servo library provides easy-to-use functions for controlling servo motors. The code executed without errors.
- Timing and Delays: We noticed that the timing and delay settings in the code can affect the speed and smoothness of the servo's movement. Adjusting these parameters allowed us to fine-tune the motor's performance.

**Conclusion**

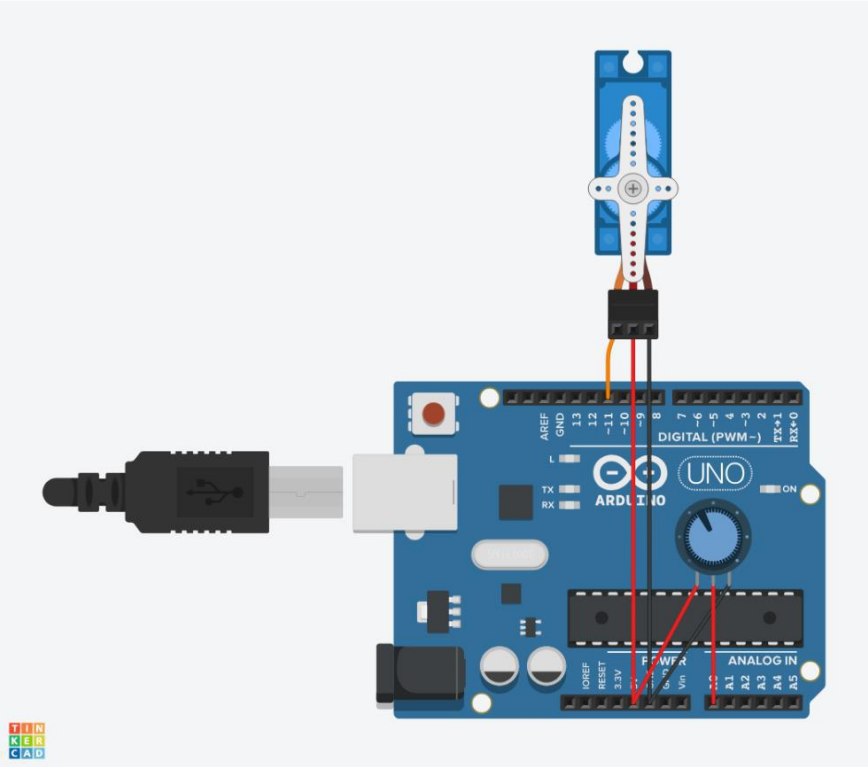
In Activity 1, we successfully controlled a servo motor to move from 0 to 180 degrees and back using Arduino. We learned that servo motors are precise and reliable devices for controlling angular motion, making them ideal for applications requiring accurate positioning.

## Activity 2: Controlling Servo with a Potentiometer

### Expected Behavior

In this activity, we expect the servo motor's position to be adjustable by manually turning the potentiometer. As we rotate the potentiometer, the servo should follow, changing its position accordingly.

### Circuit Diagram:




### Arduino Code:

```
// Exiquiel John A. Pines
// Controlling Servo with a Potentiometer

#include <Servo.h>

Servo myservo;
int potPin = A0;
int val;

void setup()
{
  myservo.attach(11);
}

void loop()
```

```
{
  val = analogRead(potPin);
  int angle = map(val, 0, 1023, 0, 180);
  myservo.write(angle);
  delay(10);
}
```

**Observations and Challenges**

- Potentiometer Control: The potentiometer allowed us to control the servo motor's position smoothly. As we turned the potentiometer, the servo responded in real-time.
- Analog Input: We observed that the potentiometer's middle pin, connected to analog pin A0 on the Arduino, acted as a voltage divider. The analogRead function provided values corresponding to the potentiometer's position.
- Code Modification: Adapting the code for this activity was manageable. We read the analog input from the potentiometer and used it to set the servo motor's position.
- Wiring: Proper wiring was crucial for this activity. Connecting the potentiometer to the 5V and GND pins and the middle pin to A0 was essential for the circuit to function correctly.

**Conclusion**

In Activity 2, we learned how to control a servo motor's position dynamically using a potentiometer. This demonstrates the versatility of servo motors in responding to real-time input, making them suitable for applications where continuous adjustments are necessary.

**Summary**

In this lab, we explored the control of servo motors using an Arduino microcontroller. We successfully implemented two activities, demonstrating the precise and versatile nature of servo motor control. This knowledge can be applied in various projects, including robotics, automation, and model control systems, where precise angular positioning is required. Understanding the interaction between servo motors and input devices, such as potentiometers, is fundamental for designing systems that involve real-time adjustments. Overall, this lab provided valuable hands-on experience in working with servo motors and Arduino-based control systems.