REPUBLIC OF THE PHILIPPINES

SURIGAO DEL NORTE STATE UNIVERSITY

Narcisso Street, Surigao City 8400, Philippines

Caspe,Mark Angelo J

BSECE-3A

Laboratory Activity no.4

Single Servo:Servos

**Introduction:**

Spinning a motor is good fun but when it comes to projects where motion control is required they tend to leave us wanting more. The answer? Hobby servos. They are mass produced, widely available and cost anything from a couple of dollars to hundreds. Inside is a small gearbox (to make the movement more powerful) and some electronics (to make it easier to control). A standard servo is positionable from 0 to 180 degrees. Positioning is controlled through a timed pulse, between 1.25 milliseconds (0 degrees) and 1.75 milliseconds (180 degrees) (1.5 milliseconds for 90 degrees). Timing varies between manufacturer. If the pulse is sent every 25-50 milliseconds the servo will run smoothly. .

**Objectives:**

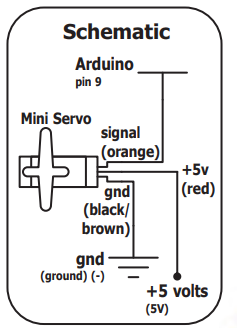
The objective of using Proteus to simulate the behavior of hobby servos in motion control projects is to test and refine control algorithms, understand system behavior, and ensure that the project functions as intended before implementing it physically. This helps in developing efficient and reliable motion control systems.

**Materials:**

1.) Atmega328P 2.) 22pf Capacitor 3.) Push Button

4.) Crystal 5.) Mini Servo x1

**Schematic Diagram:**



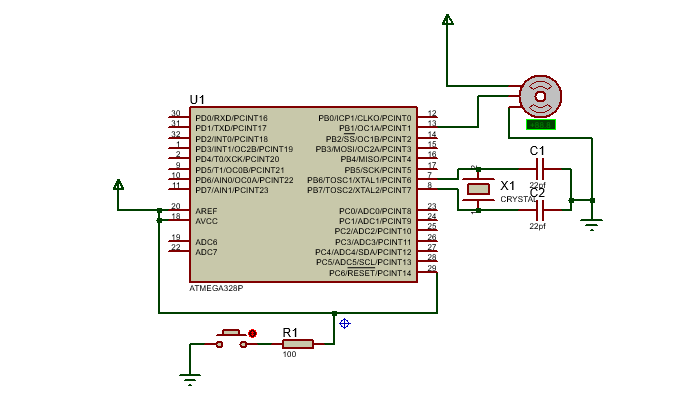
**Procedure:**

* Open Proteus and Create New Project: Launch Proteus and start a new project.
* Find Components: Locate the components needed for the simulation. This includes the hobby servo, microcontroller (e.g., ATmega328P), and any additional components required for your project.
* Refer to Schematic Diagram: Review the schematic diagram to understand how the wires are connected. Ensure that the connections match the specifications of your project.
* Write Code in Arduino IDE: Open the Arduino IDE and write the code for controlling the servo. Save the code for later use.Save Code File: Save the code file in a location where it's easily accessible.Return to Proteus:
* Go back to Proteus and click on the microcontroller component (e.g., ATmega328P).Select Code File: In the microcontroller properties, locate the option to select the code file (usually a .hex file). Browse for the .hex file you saved earlier and select it.
* Simulate: Run the simulation in Proteus to see how the hobby servo responds to the signals generated by the microcontroller.

**Results and Discussion:**

In the simulation, the servo motor undergoes a continuous rotation from 0 to -90 degrees and then from -90 back to 90 degrees, repeating this cycle. By adjusting the code, you have the ability to control the servo motor's movement according to your specified references or parameters. This allows for precise manipulation and customization of the servo motor's behavior to suit your needs.

**Circuit:**

****

**Program:**

#include <Servo.h>

Servo myservo;

int pos = 0;

void setup() {

myservo.attach(9);

}

void loop() {

for(pos = 0; pos < 180; pos += 1)

{

myservo.write(pos);

delay(15);

}

for(pos = 180; pos>=1; pos-=1)

{

myservo.write(pos);

delay(15);

}

}

**Findings:**

The servo motor swings back and forth between 0 and -90 degrees, then reverses to 90 degrees, repeating this motion. By tweaking the code, you can control how the motor moves, adjusting its position and speed based on what you need it to do. This flexibility allows you to use the motor in different ways, making it useful for all sorts of projects.

**Recommendations:**

### I recommend you can interact with the code to play and control the servo motor's movement. By adjusting specific parts of the code, you can customize the motor's behavior to match your desired angle or position. This allows you to understand which sections of the code influence the motor's movement, empowering you to tailor it to your specific needs or references.

**Conclusions:**

By playing with the code in the simulation, you can control the servo motor's movement and customize it to your needs.