Prerequisites

This worksheet builds upon the theory of how to use an external pin that was covered in the Lab 2 worksheet. Whilst you won't need any code from Lab 2, you will need to have covered how to use an electronic breadboard and how to wire up an external output to an Arduino pin. Out of your Arduino pack, you will need: 1 Arduino, 1 Servo, and 3 jumper cables.

Objectives

The objective of today's lab is to learn how to control a servo using the inbuilt Arduino Servo Library: `https://docs.arduino.cc/libraries/servo/`.

This worksheet will take you through the steps of how to connect an external servo to your Arduino and create a sketch that rotates the servo in increments of 30 degrees. Once the sketch reaches 180 degree (the maximum number of degrees on a servo), the sketch will reset the degrees to zero and start incrementing in steps of 30 degrees again.

Understanding how a servo works

Within the remit of this lab, a servo (or servomotor) is a physical device used to control rotary motion. On an Arduino, servos (and some other analogue devices) need to be controlled using Pulse Width Modulation (PWM). Whilst this lab will not go into detail about PWM (the interested reader can view `https://docs.arduino.cc/learn/microcontrollers/analog-output/`), it is important to understand that not every pin on an Arduino can be used for PWM output. On the Arduino Nano Every, the pins that can handle a PWM output are pins: 3, 5, 6, 9, 10 (these details can be viewed from the datasheet linked in lecture 1.)

On an Arduino there is an inbuilt Servo library that encapsulates the PWM complexities and helps us control a servo (or an Electronic Speed Controller connected to a DC motor). For this library, you simply write an output value to your servo between 0 and 180 — which maps directly to the number of degrees of rotation available on most servos: 0→180.

A servo has three wires coming out of it: a *power* cable, a *ground* cable, and a *signal* cable. The power cable is usually red, the ground cable is usually black or brown, and the signal cable is usually white or orange.

1 Creating the circuit

The circuit needed for this worksheet is: Arduino Pin 9→Servo sensor cable, Arduino GND→Servo ground cable, and Arduino +5V→Servo power cable.

2 Setup the servo library

The first thing we need to do is create a blank sketch, this is done by opening Arduino IDE, going to 'File'→'Examples'→'01.Basics'→'BareMinimum'. Once this has opened, we can now setup the Servo library.

In the global scope (the top of the file), enter in the code below. This code firstly links our sketch to the servo library, then defines a new named pin called '$SERVO\_PIN$' on pin 9, and then creates an instance of the servo library for us to interact with called 'servo'.

```
#include <Servo.h>
#define SERVO_PIN 9
Servo servo;
```

The next step is for us to declare the array (vector) of integer values we want the sketch to output. For simplicity, we will say for the time being that should be every 30 degrees from 0 to 180. The code below needs to be put into the global scope below the previous code. This code operates on the same logic as Lab 1 and Lab 2; however, instead of having an array of dots and dashes, we now have an array of degrees to rotate the servo to.

```
int output[7] = { 0, 30, 60, 90, 120, 150, 180 };
int output_position = 0;
```

3 Setup function

In the setup function of the sketch, we need to do three things: 1 - setup the standard debugging tools, 2 - set the position in the output, 3 - set the output pin and attach it to the servo library. The code in the following sections, needs to go into the 'setup' function.

3.1 Setup debugging

The first thing to do in any sketch, is to setup the debugging by connecting the Serial. This allows us to monitor what the Arduino is doing via displaying specific messages at certain points throughout the sketch. This code attaches the serial, pauses the system for 3 seconds, and displays

a message 'System Ready'.

```
Serial.begin(9600);
delay(3000);
Serial.println("System Ready");
```

3.2 Setup output

In the global scope, we have already declared the desired output; however, it is always good practice to set any variables that could be changed at different points during the sketch. In our case, this is simply means setting the current position in the degree output.

```
output_position = 0;
```

3.3 Setup servo

In the global scope, we have setup the servo library, but we now need to configure the servo library to talk to the correct pins. In our case this is pin 9 that we previously named '$SERVO\_PIN$'. The first thing we need to do is set pin 9 to be an output pin. We can then attach pin 9 to the servo library. Finally, we can set the servo to 0 degrees.

```
pinMode(SERVO_PIN, OUTPUT);
servo.attach(SERVO_PIN);
servo.write(0);
```

4 Loop function

Now that we have the setup function sorted, we can concentrate on the loop function. Within the loop function, there are three major steps: 1 - write the correct value to the servo, 2 - determine the next position in the 'output' array, 3 - wait two seconds before the next iteration. The code in the following sections, needs to go into the 'loop' function.

4.1 Output to servo

At the top of the 'loop' function, needs to be the logic to write the correct value within the output to the servo. Thankfully, this is one line of code. We need to access the current position within the output array and then tell the servo to use that value. We also add in an additional line here, to output to the serial connection the selected value so that we can monitor the progress of the sketch.

```
servo.write(output[output_position]);
Serial.println(output[output_position]);
```

4.2 Determine next position

The next logic within the 'loop' function, is the logic to determine the next position within the output array. Thankfully, this is the same logic that is used to determine the next position within the dots and dashes array in Lab 1.

```
if (output_position >= 6) {
  output_position = 0;
} else {
  output_position++;
}
```

4.3 Wait

The final logic within the 'loop' function, is to wait two seconds until the next iteration. This is done by telling the Arduino to wait 2000 milliseconds.

```
delay(2000);
```

Try it out on your Arduino

You now should be able to compile your sketch ('Sketch'→'Verify/Compile') and then upload to your Arduino ('Sketch'→'Upload'). You should see that your Arduino now rotates your servo 30 degrees every 2 seconds until it gets to 180, at which point it restarts back from 0.

Once you have a working sketch from the above code, expand the sketch to rotate in 30 degree increments from 0 to 180 and then rotate in 30 degree decrements from 180 back down to 0 without repeating two identical degree values in a row.

Help

If you are having issue compiling the project or don't understand where each section goes, you can see a complete version of the sketch with additional source code comments at `https://github.com/dsbrennan/mac-233-arduino-labs/blob/main/lab-3-servo/lab-3-servo.ino`.