Lab 3 - Drive a Motor

Adapted tutorial from randomnerdtutorials.com https://randomnerdtutorials.com/getting-started-with-esp32/

## Learning Outcomes:

* Learn about servos/pulse width modulation
* Learn about voltage regulators
* Learn how to spin a servo
* Spin a servo using Bluetooth

## Background:

A servo is a low-speed high torque motor. They are simple to use. Most of these servos are useful because they can determine their angular position using a potentiometer ([link](https://en.wikipedia.org/wiki/Potentiometer)). These cheap lightweight control options are used in robotic and hobby aircraft applications where position is a concern. For example, to control aircraft control surfaces. The downside to a potentiometer servo is that it can’t spin more than a certain angle (usually 180deg). The servos that we’ll use are called continuous rotation servos. They remove the potentiometer and essentially operate the same as a motor. However, that comes at the cost of not being able to determine angular position.



*9g servo. Image courtesy of* [*Amazon*](https://www.amazon.com/Tower-Pro-SG90-Analog-Servo/dp/B07B8SJQJD/ref=asc_df_B07B8SJQJD/?tag=hyprod-20&linkCode=df0&hvadid=278525945678&hvpos=&hvnetw=g&hvrand=5411674091295809978&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9029858&hvtargid=pla-605972587590&psc=1)*.*

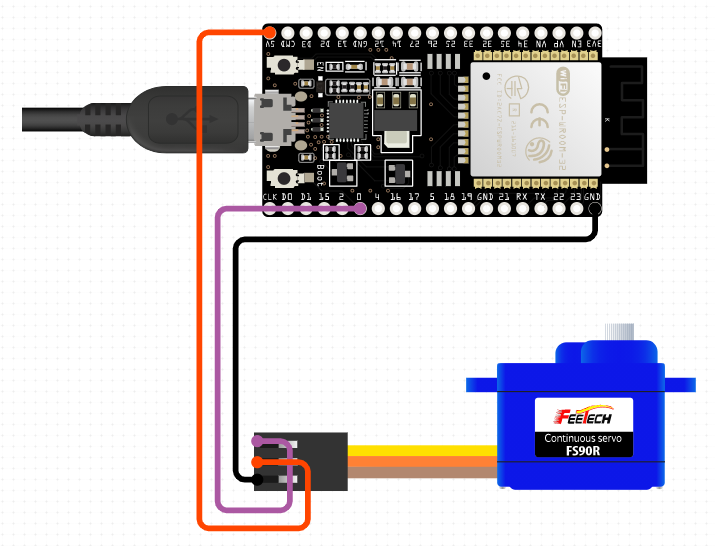
These servos are simple to use which is why they are used in many applications including hobby aircraft, robotics, drones, etc. We will be using these as the motors for our AutonoMouse. To operate the servos we only need 3 wires per servo. 1) black for ground, 2) red for 5V power, and 3) orange for a signal (this signal is called pulse-width modulation [Pulse-width modulation](https://en.wikipedia.org/wiki/Pulse-width_modulation), [Servo control](https://en.wikipedia.org/wiki/Servo_control#:~:text=Servo%20control%20is%20a%20method,less%20common%20today)).

The objectives for today’s lab are to 1) power the servo with power and a signal from the Arduino Nano, 2) power the servo via external power source, and 3) power the servo via external power source and connect via Bluetooth.

## Step 1 Power Servo from ESP32:

We will first test our servo using the 5V pin from our esp32. This is not good for high current applications because running too much current through the esp32 can break it, so don’t put too much torque on the servo or run it for too long. For a simple test to make it move clockwise and counterclockwise it’ll be fine.

1. Create the circuit shown below
   1. Connect 5V to red
   2. Connect gnd to brown
   3. Connect Pin0 to orange
2. Copy/paste the Lab3-2.ino into Arduino IDE and upload to your Arduino Nano
3. Watch the servo spin clockwise
4. Comment out the first void loop() in the code and uncomment the second (you comment things by adding ‘//’ in front the line of code)
5. Upload your code again and watch your servo spin the other direction.
6. BONUS QUESTION: Continuous servos are controlled in Arduino using the Servo.h library. 0 makes the servo spin fastest in the clockwise direction and 180 makes the servo spin fastest in the counterclockwise direction. 90 makes the servo not spin. (These values might change slightly depending on your servo). Play around with these values to make them spin slower or faster.



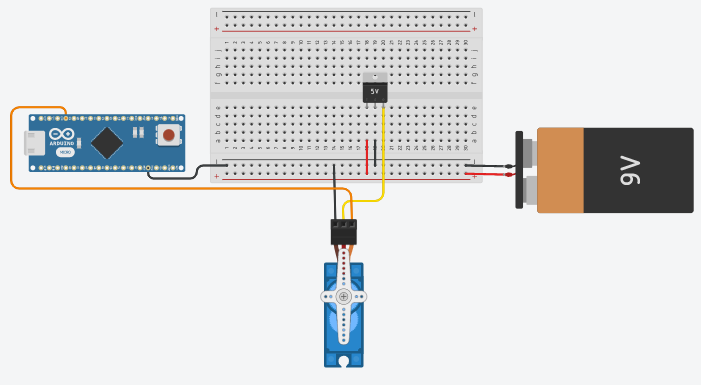
*Circuit created in circuito.io*

## Part 2 Power Servo using a Voltage Regulator:

We can power our servo using an external power source. For this part we will use a 9V. However, running 9 volts (V) through a 5V rated servo would damage it. To decrease the voltage, we are going to use a voltage regulator. A voltage regulator ensures that the voltage doesn’t increase past a certain level. Different regulators are made for different voltage levels. You might have a 3.3, 5V, etc. We will pass the 9V current from the battery through a 5V voltage regulator, which will output 5V, to spin the servos.

The voltage regulator we’re using has 3 pins. 1 for output, 2 for ground, and 3 for input voltage. When working with voltage regulators make sure to consult their datasheets to make sure not to use them improperly. If you pass too high a current through a voltage regulator you can ruin it. Another thing that to remember with voltage regulators is that they can’t increase voltage. The input voltage must be the same or lower than the desired output voltage. They also will get hot after being used for some time.

1. Create the circuit shown below (substitute the esp32 for the Arduino Nano, connect the signal wire to pin0 on the esp32)
2. Use the same code as before (the signal wire, orange colored wire, from the Arduino Nano is the same we’re just powering it with something else now)
3. Do the same thing as before. Spin the servo clockwise, spin the servo counterclockwise)
4. BONUS QUESTION: how much current can the AN7805 voltage regulator withstand. (Lookup its datasheet online)



*Circuit created in Tinkercad*

## Part 3 Control a Servo via BT:

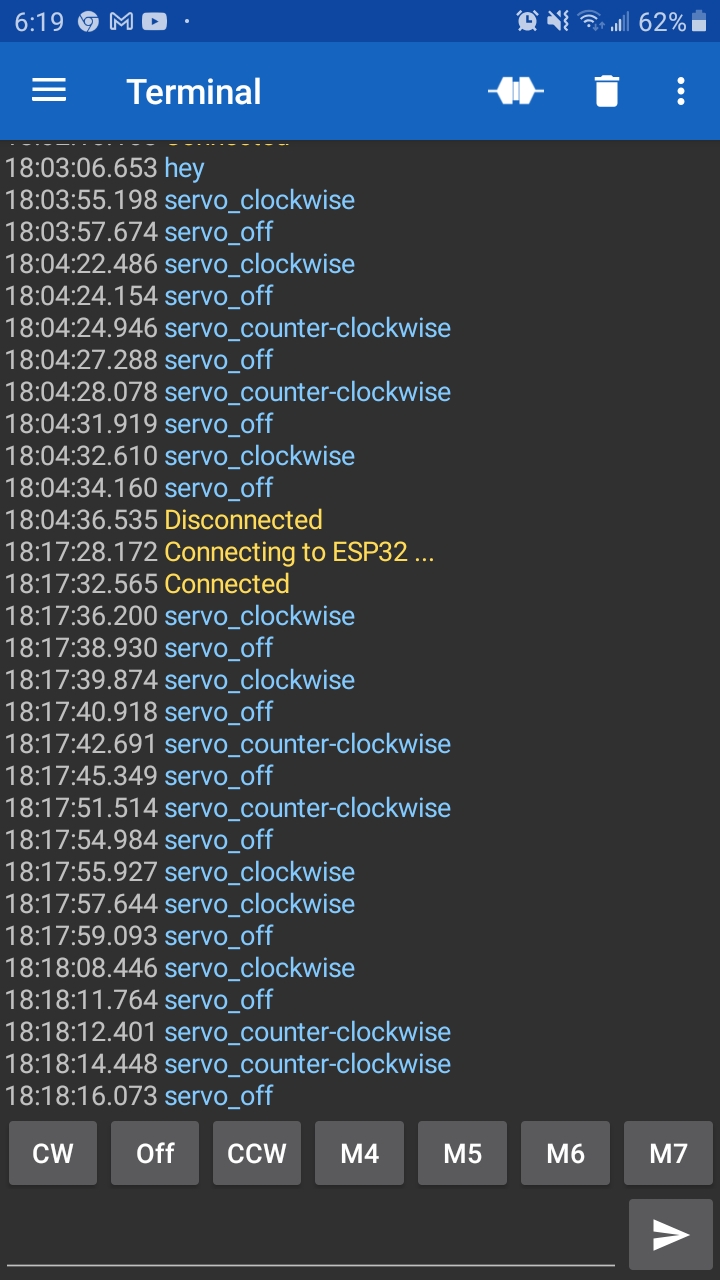
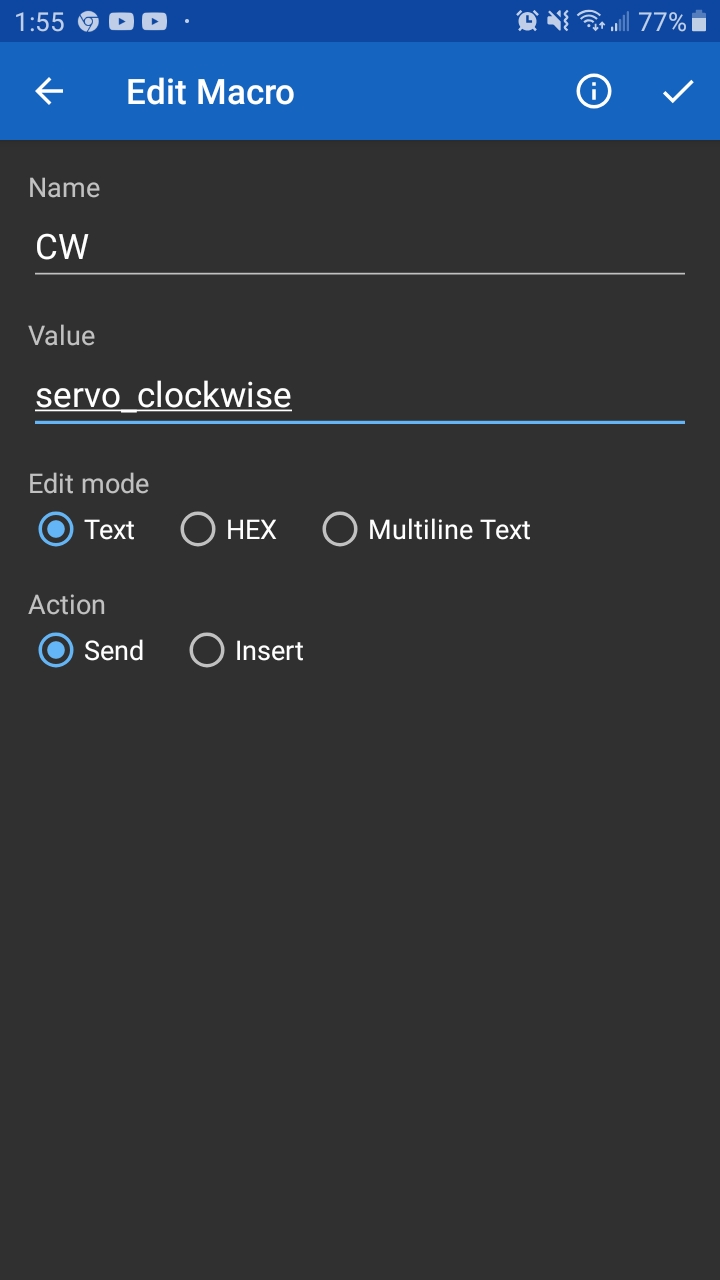
To finish this lab, we will now control our servo circuit from part 2 using our Bluetooth app from the last lab.

First, make the circuit from part 2.

Next, download, paste, and upload Lab3-1.ino provided in this lab to the esp32 using Arduino IDE.

Next, use your phone serial Bluetooth app to connect to the esp32 like the last lab. You should receive a similar message as last time: “The device started now you can pair it with Bluetooth!”.

Lastly, create 3 buttons to send a text object. Make them send text variables labeled “servo\_clockwise”, “servo\_counter-clockwise”, and “servo\_off”. Double check that your spelling is correct otherwise it won’t work.



Test your Bluetooth connection by turning your servo on and off using your phone. You can also survey what is being outputted by using Arduino IDE’s serial monitor. Make sure that the baud rate is set to 115200.