·LESSON 04·

Motor Control: Part I

Lesson Overview

We shall now learn the basics of the devices that move our robotic arm: Motors. Since real motors cannot be used, we shall focus on the correct ways to implement different kinds of motors and their driver circuits.

The topics covered in part I are:

- Brushed DC motors
- Transistor motor driver circuits
- Servo motors

Brushed Motors

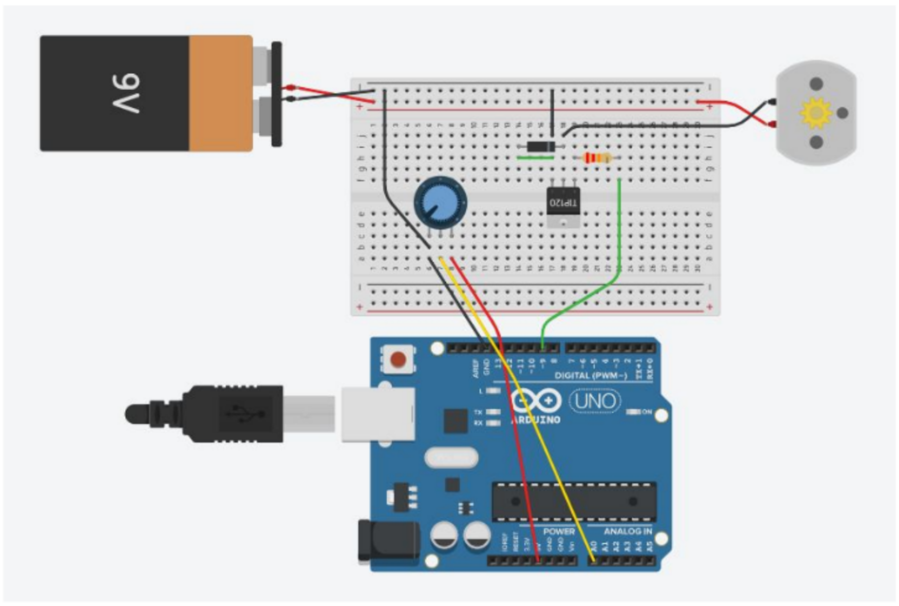
DC brushed motors are the cheapest and most common type of motor. It has two terminals, which when connected to the terminals of a battery, cause the motor to rotate. Switching polarity reverses the direction of rotation.

The maximum current you can draw out of an Arduino from its pins is around 40mA; such a small amount of current is not sufficient to drive a motor. The currents required to drive a motor are typically 0.5-2A or more, and driving such large currents from Arduino may fry the board. An intermediate circuit called a **motor driver** is used to handle these large currents.

The next example uses a transistor to drive a DC motor.

DC Motor Control Using A Transistor

The usual BC547 NPN transistor we have been using till now cannot handle currents high enough to drive motors. Instead, we shall use the TIP120 darlington transistor. It can handle upto 5A collector current (the BC547 can only handle 100mA).
The TIP120 acts like a high power switch for the motor. When an Arduino pin gives it a high signal, it switches on the motor. To control the motor's speed, analogWrite() can be used. The following circuit includes the driving components and a potentiometer to control the speed.

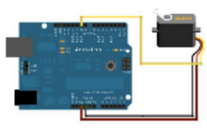


The code can be found here. You may find this schematic easier to read. Note, the diode in parallel to the motor is to prevent high voltage spikes from ruining the transistor. For more about the TIP120, check out this tutorial.

Servo Motors

Servo motors, unlike DC motors, can be positioned to a specific angular position. They are widely used in robotics, especially in devices like the robotic arm.

These motors have just 3 pins - *power*, *ground*, and *signal*. The Arduino can control the angular position of the servo by sending it a specific time-coded PWM signal through this SIGNAL pin. Fortunately, we don't have to worry about the details, since the `Servo.h` library takes care of that. The circuit in the following tutorial controls the position of a servo according to a potentiometer's input. Try to implement it in TinkerCAD.



Tutorial
Arduino: Knob
https://www.arduino.cc/en/Tutorial/Knob

To learn more about servo motors, including how they read PWM signals and their inner working, read this tutorial.