Adafruit's Raspberry Pi Lesson 9. Controlling a DC Motor

Created by Simon Monk
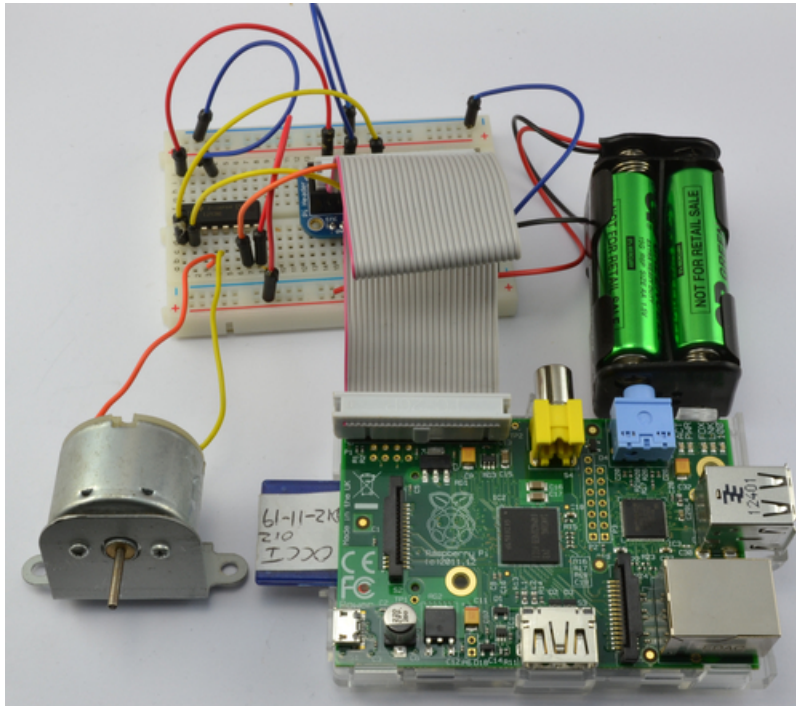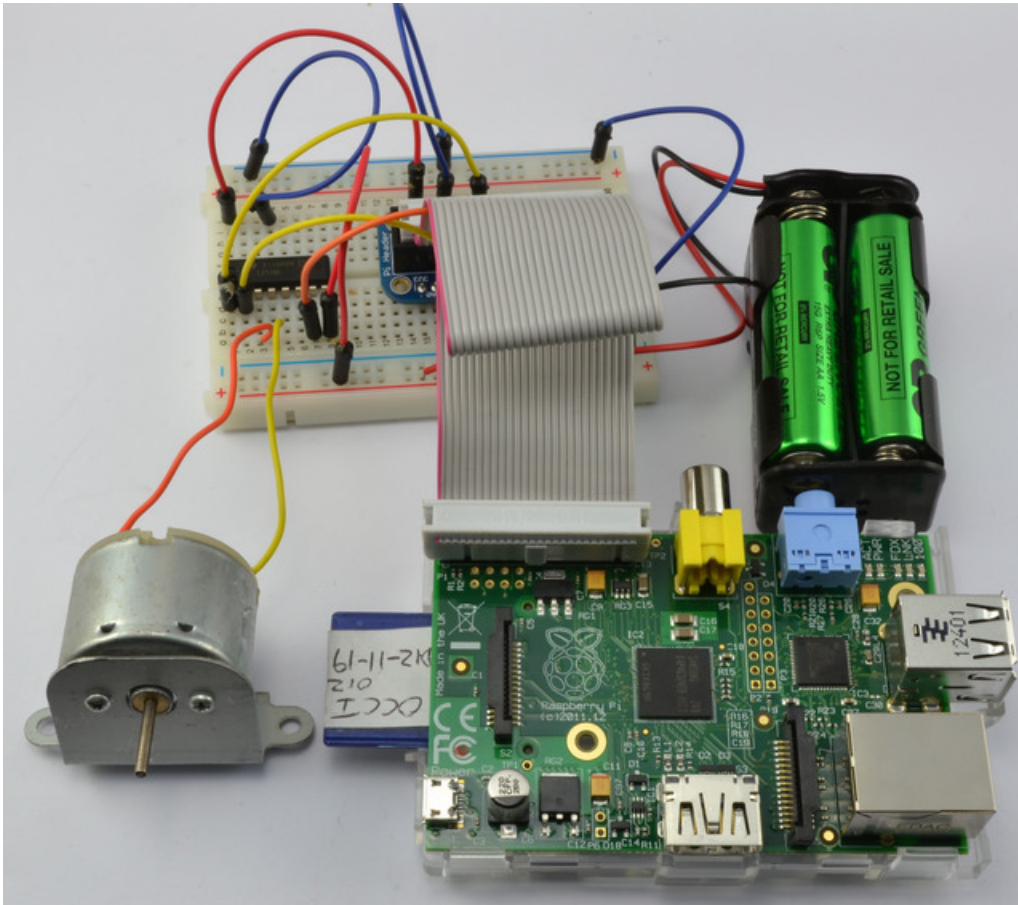
Guide Contents

Overview

This lesson describes how to control both the speed and direction of a DC motor using Python and a L293D chip.

In Lesson 8, we used the Pi to generate pulses to control the position of a servo motor. In this lesson we use pulses to control the speed of a regular DC motor and the L293D motor control chip to reverse the direction of the current through the motor and hence the direction in which it turns.

Parts

To build this project, you will need the following parts.

| Part	Qty
Raspberry Pi	1
Cobbler Breakout Board	1
Set of male to male jumper leads	1
Half-size breadboard	1
6V DC Motor	1
L293D Motor Controller IC	1
Adafruit Occidentalis 0.2 or later operating system distribution.	1
4 x AA or AAA battery holder and batteries.	1

PWM

Pulse Width Modulation (or PWM) is a technique for controlling power. We use it here to control the amount of power going to the motor and hence how fast it spins.

The diagram below shows the signal from the PWM pin of the Raspberry Pi.
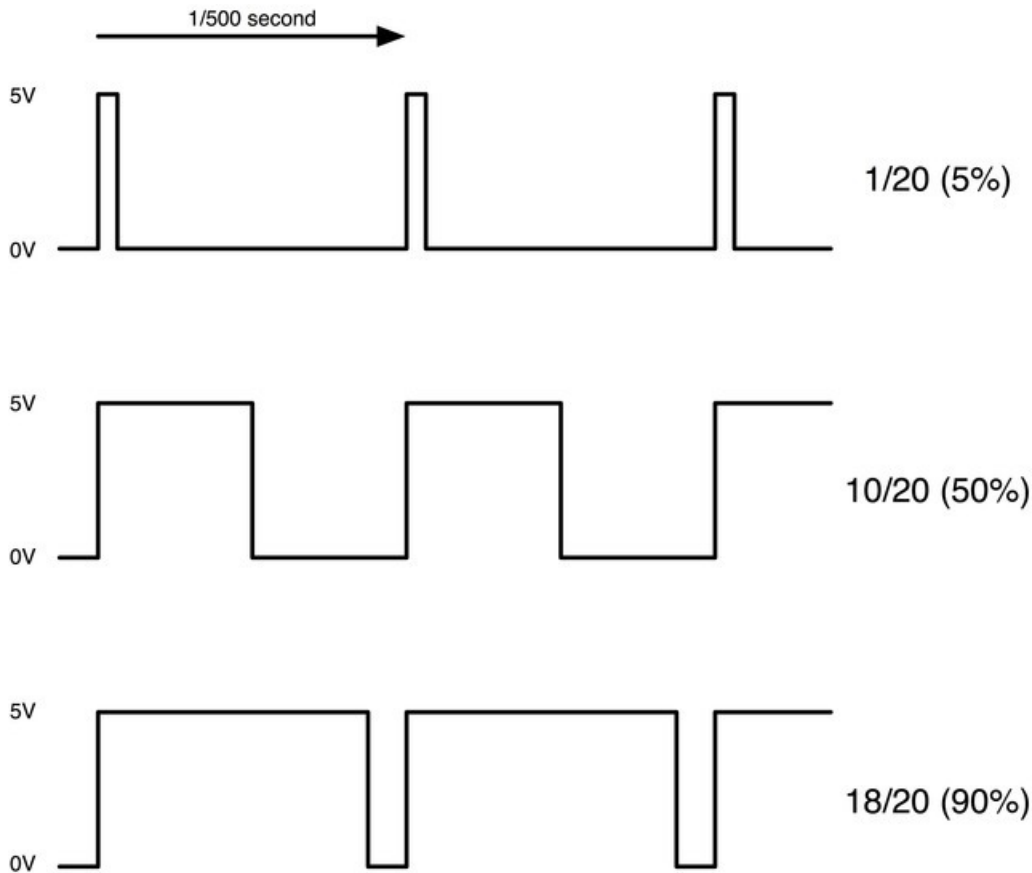
Every 1/500 of a second, the PWM output will produce a pulse. The length of this pulse controls the amount of energy that the motor gets. No pulse at all and the motor will not turn, a short pulse and it will turn slowly. If the pulse is active for half the time, then the motor will receive half the power it would if the pulse stayed high until the next pulse came along.

The PWM Kernel Module

Adafruit and Sean Cross have created a kernal module that is included with the Occidentalis (https://adafru.it/aQx) distribution. For details of creating an Occidentalis follow this link (https://adafru.it/aWq). If you want to use the module with Raspbian or some other distribution, then there is help on installing the kernel module into your environment here (https://adafru.it/aQx).

You used the PWM and Servo Kernel Module in Lesson 8, to control a Servo. This time, you will use the same module to control the speed of the motor.
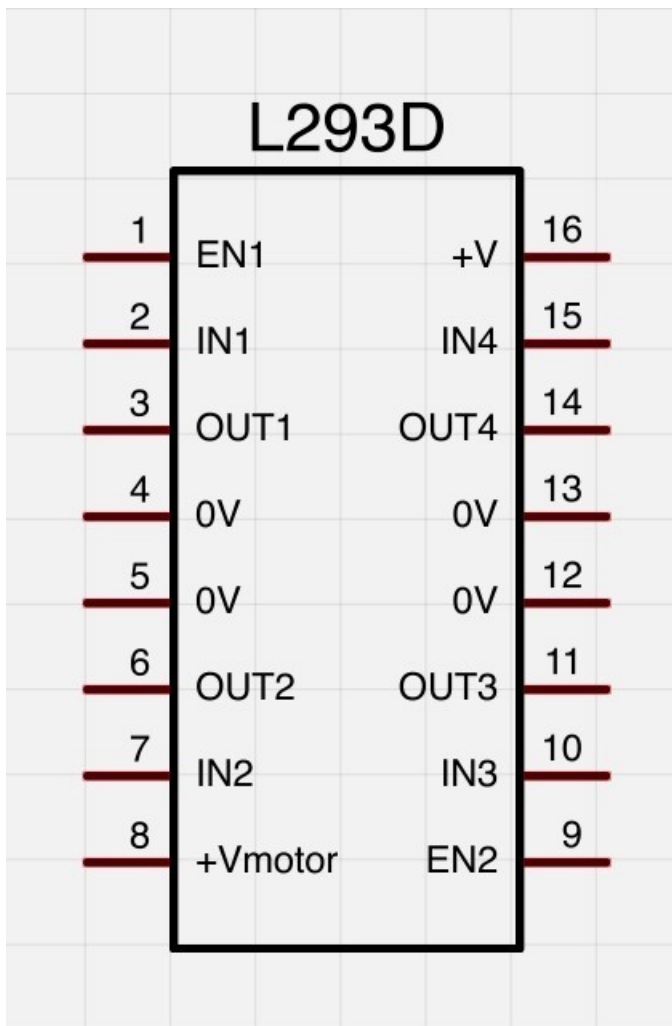
The modules interface uses a file type of interface, where you control what the output pin and therefore the servo is doing, by reading and writing to special files. This makes it really easy to interface with in Python or other languages.

The files involved in using the module to drive a servo are listed below. All the files can be found in the directory /sys/class/rpi-pwm/pwm0/

File	Description
active	This will be 1 for active 0 for inactive. You can read it to find out if the output pin is active or write it to make it active or inactive.
delayed	If this is set to 1, then any changes that you make to the other files will have no effect until you use the file above to make the output active
mode	Write to this file to set the mode of the pin to either pwm, servo or audio. Obviously we want it to be pwm. Note that the pin is also used by the Pi's audio output, so you cannot use sound at the same time as controlling a motor.
frequency	This is the number of pulses to be produced per second.
duty	The value here needs to be between 0 and 100 and represents the percentage of the pulse during which the power to the motor is on. The higher the number, the faster the motor will spin.

L293D

This is a very useful chip. It can actually control two motors independently. We are just using half the chip in this lesson, most of the pins on the right hand side of the chip are for controlling a second motor, but with the Raspberry Pi, we only have one PWM output.

```
              L293D
   1  ┌─────────────────┐  16
──────┤ EN1          +V ├──────
   2  │                 │  15
──────┤ IN1         IN4 ├──────
   3  │                 │  14
──────┤ OUT1       OUT4 ├──────
   4  │                 │  13
──────┤ 0V           0V ├──────
   5  │                 │  12
──────┤ 0V           0V ├──────
   6  │                 │  11
──────┤ OUT2       OUT3 ├──────
   7  │                 │  10
──────┤ IN2         IN3 ├──────
   8  │                 │  9
──────┤ +Vmotor     EN2 ├──────
      └─────────────────┘
```

The L293D has two +V pins (8 and 16). The pin '+Vmotor (8) provides the power for the motors, and +V (16) for the chip's logic. We have connected pin 16 to the 5V pin of the Pi and pin 8 to a battery pack.

Hardware

There are two reasons why we need to use a L293D chip in this project. The first is that the output of the Raspberry Pi is nowhere near strong enough to drive a motor directly and to try this may damage your Raspberry Pi.

Secondly, in this lesson, we want to control the direction of the motor as well as its speed. This is only possible by reversing the direction of the current through the motor, something that the L293D is designed to do, with the help of two control pins.
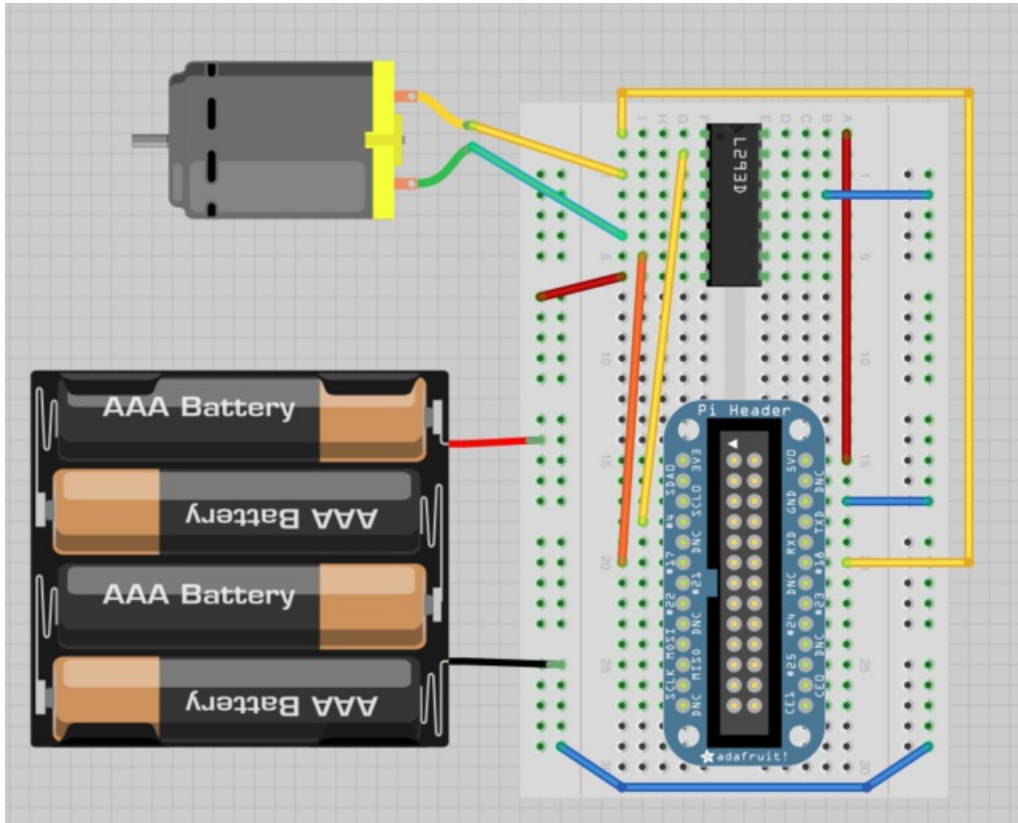
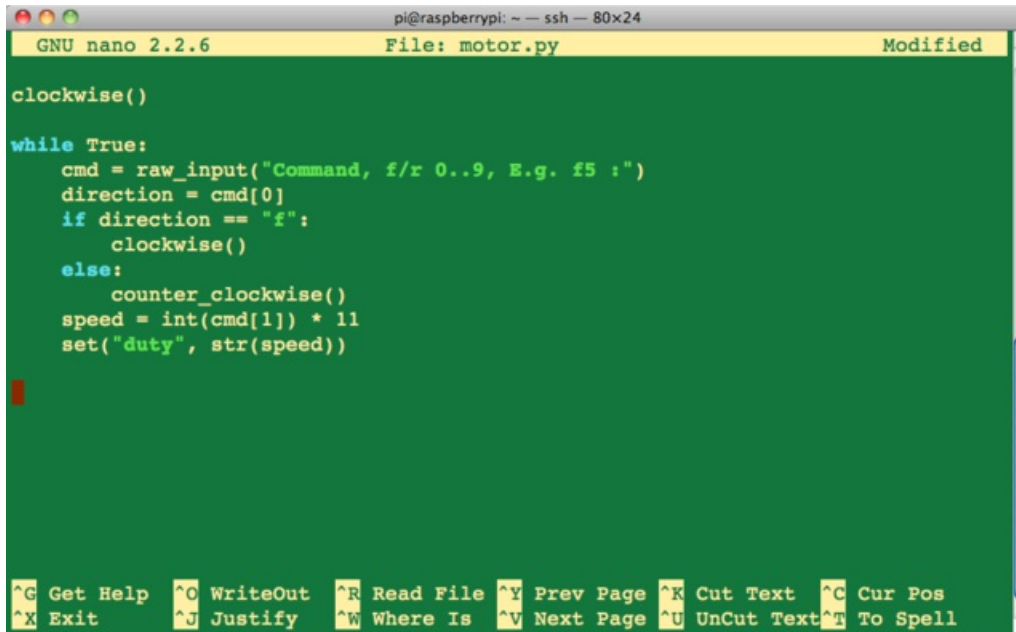The project fits nicely on a half-sized breadboard.

Software

Since we need to control pins (connected to pin 4 and 17 of the GPIO) we also need to use the GPIO library. For details of this, see Lesson 4 (https://adafru.it/aTH).

There are lots of ways to get the sketch from the listing below onto your Raspberry Pi. Perhaps the easiest is to connect to your Pi using SSH (See Lesson 6 (https://adafru.it/aWc)) opening an editor using the command below:

```
nano motor.py
```

and then pasting in the code below, before saving the files using CTRL-x.

Here is the code:

```
import RPi.GPIO as io
io.setmode(io.BCM)

in1_pin = 4
in2_pin = 17

io.setup(in1_pin, io.OUT)
io.setup(in2_pin, io.OUT)

def set(property, value):
    try:
        f = open("/sys/class/rpi-pwm/pwm0/" + property, 'w')
        f.write(value)
        f.close()
    except:
        print("Error writing to: " + property + " value: " + value)

set("delayed", "0")
set("mode", "pwm")
set("frequency", "500")
set("active", "1")

def clockwise():
    io.output(in1_pin, True)
    io.output(in2_pin, False)

def counter_clockwise():
    io.output(in1_pin, False)
    io.output(in2_pin, True)

clockwise()

while True:
    cmd = raw_input("Command, f/r 0..9, E.g. f5 :")
    direction = cmd[0]
    if direction == "f":
        clockwise()
    else:
        counter_clockwise()
    speed = int(cmd[1]) * 11
    set("duty", str(speed))
```

The Python program first sets-up the two GPIO pins to be outputs. It then defines the same convenience function ("set") that we used in Lesson 8, to write to the PWM Kernel Module. This is then used to set the parameters for PWM.

There are two other functions defined, "clockwise" and "counter_clockwise", that control the direction of the motor by changing the two input pins.

If both control pins are HIGH, or both LOW then the motor will be stopped. But if IN1 is HIGH and IN2 LOW it rotates in one direction, reversing in direction if the values of IN1 and IN2 are reversed.

The main loop of the program waits for you to enter commands in the format of a letter ("f" or "r") and a digit between 0 and 9. The letter sets the direction of the motor and the digit the speed, by multiplying the digit by 11 to give a value between 0 and 99 that the PWM library expects.

Test & Configure

To run the program, you will need to run it as super user to get access to the GPIO pins, so type:

```
sudo python motor.py
```

You will then get a prompt for you to enter a command. Try entering a few 2 letter commands as shown below.

```
$sudo python motor.py
Command, f/r 0..9, E.g. f5 :f9
Command, f/r 0..9, E.g. f5 :r4
Command, f/r 0..9, E.g. f5 :
```