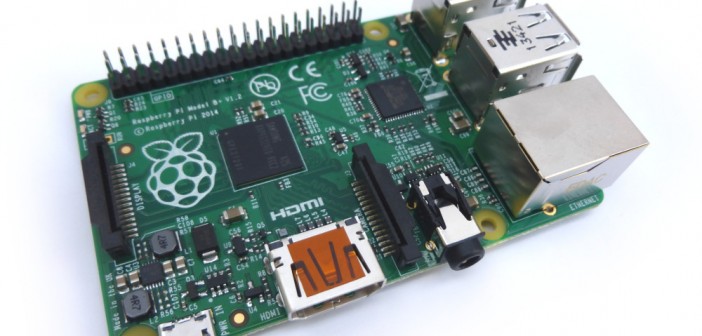
Stepper Motor Control In Python

[55](https://www.raspberrypi-spy.co.uk/2012/07/stepper-motor-control-in-python/#comments)

BY [MATT](https://www.raspberrypi-spy.co.uk/author/matt/) ON JULY 11, 2012[ROBOTICS](https://www.raspberrypi-spy.co.uk/category/hardware/robotics/), [TUTORIALS & HELP](https://www.raspberrypi-spy.co.uk/category/tutorials-help/)

[](https://www.raspberrypi-spy.co.uk/wp-content/uploads/2013/06/default_featured.jpg)

Having played with LEDs, switches and buzzers I felt the natural next step was playing with a stepper motor or two. Unlike conventional electric motors, stepper motors allow you to rotate the axis in precise increments. This makes them useful in all sorts of Raspberry Pi projects.

**Basic Stepper Motor**



There is a huge selection of stepper motors to buy but I decided to experiment with a 28BJY-48 with ULN2003 control board. The reasons I chose this device where :

* It is cheap
* Runs on 5V
* Easy to interface to the Pi’s GPIO header
* Small but relatively powerful
* Widely available from both overseas and UK sellers
* Easy to obtain with a controller board

There are additional details in the [Stepper Motor 28BJY-48 Datasheet](https://www.raspberrypi-spy.co.uk/wp-content/uploads/2012/07/Stepper-Motor-28BJY-48-Datasheet.pdf).

**Buy Stepper Motors for the Pi**

The 28BJY-48 stepper motors can be obtained from :

* [eBay](https://rover.ebay.com/rover/1/710-53481-19255-0/1?mpre=https%3A%2F%2Fwww.ebay.co.uk%2Fsch%2Fi.html%3F_odkw%3Dstepper%2Bmotor%26_osacat%3D0%26_from%3DR40%26_trksid%3Dp2045573.m570.l1313.TR11.TRC1.A0.H0.X5V%2Bstepper%2Bmotor.TRS0%26_nkw%3D5V%2Bstepper%2Bmotor%26_sacat%3D0&campid=5337928941&toolid=20008)
* [Amazon](http://www.amazon.co.uk/s/?field-keywords=5v%20stepper%20motor&tag=raspberrypispy-21)

**Interfacing With The Pi**

[](https://www.raspberrypi-spy.co.uk/wp-content/uploads/2012/07/28BJY-48-Stepper-Motor.jpg)

The stepper motor connects to the controller board with a pre-supplied connector. The controller board has six pins which need to be connected to the Pi’s GPIO header :

* 5V (P1-02)
* GND (P1-06)

and

* Inp1 (P1-11)
* Inp2 (P1-15)
* Inp3 (P1-16)
* Inp4 (P1-18)

The P1-XX references above represent the [Pi header](https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/) pins I used. These are defined in the Python example below in the StepPins list so if you use different pins be sure to update the Python list as well. You can use other GPIO pins if required just remember to update your Python script.

To rotate the stepper motor you provide a sequence of “high” and “low” levels to each of the 4 inputs in sequence. By setting the correct sequence of high and low levels the motor spindle will rotate. Reversing the sequence results in the direction being reversed.

**Example Stepper Motor Python Script**

Here is a copy of the [stepper motor script](https://bitbucket.org/MattHawkinsUK/rpispy-misc/raw/master/python/stepper.py" \t "_blank) I used to rotate the stepper motor. It uses the RPi.GPIO library and defines a 4-step and 8-step sequence.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | #!/usr/bin/python  # Import required libraries  import sys  import time  import RPi.GPIO as GPIO    # Use BCM GPIO references  # instead of physical pin numbers  GPIO.setmode(GPIO.BCM)    # Define GPIO signals to use  # Physical pins 11,15,16,18  # GPIO17,GPIO22,GPIO23,GPIO24  StepPins = [17,22,23,24]    # Set all pins as output  for pin in StepPins:    print "Setup pins"    GPIO.setup(pin,GPIO.OUT)    GPIO.output(pin, False)    # Define advanced sequence  # as shown in manufacturers datasheet  Seq = [[1,0,0,1],         [1,0,0,0],         [1,1,0,0],         [0,1,0,0],         [0,1,1,0],         [0,0,1,0],         [0,0,1,1],         [0,0,0,1]]    StepCount = len(Seq)  StepDir = 1 # Set to 1 or 2 for clockwise              # Set to -1 or -2 for anti-clockwise    # Read wait time from command line  if len(sys.argv)&gt;1:    WaitTime = int(sys.argv[1])/float(1000)  else:    WaitTime = 10/float(1000)    # Initialise variables  StepCounter = 0    # Start main loop  while True:      print StepCounter,    print Seq[StepCounter]      for pin in range(0, 4):      xpin = StepPins[pin]#      if Seq[StepCounter][pin]!=0:        print " Enable GPIO %i" %(xpin)        GPIO.output(xpin, True)      else:        GPIO.output(xpin, False)      StepCounter += StepDir      # If we reach the end of the sequence    # start again    if (StepCounter&gt;=StepCount):      StepCounter = 0    if (StepCounter&lt;0):      StepCounter = StepCount+StepDir      # Wait before moving on    time.sleep(WaitTime) |

You can download it directly to your Pi using :

wget https://bitbucket.org/MattHawkinsUK/rpispy-misc/raw/master/python/stepper.py

The script needs to be run using “sudo” :

sudo python stepper.py

Press Ctrl-C to quit.

**Step Wait Time**

The script adds a small delay between each step to give the motor time to catch up.In this example the default wait time is set to 0.01 seconds (10 milliseconds). To change the speed of rotation you can change this value. I found I could reduce it to 4ms before the motor stopped working. If the script runs too fast the motor controller can’t keep up. This performance may vary depending on your motor and its controller.

To specify a different wait time you can pass a number of milliseconds as an argument on the command line using :

sudo python stepper.py 20

where 20 is the number of milliseconds.

**Step Sequences**

The complete step sequence consists of 8 steps. If StepDir is set to -2 or 2 the number of steps is reduced to 4.

The 4 step sequence is faster but the torque is lower. It’s easy to stop the rotation by holding the motor spindle. The 8 step sequence is slower but the torque is much higher. For my turntable application I prefer the torque over speed so I will be using the 8 step sequence.

**Final Thoughts**

You can now control a stepper motor using a Raspberry Pi and a Python script. If you add another motor you’ve got the beginnings of a small robot!