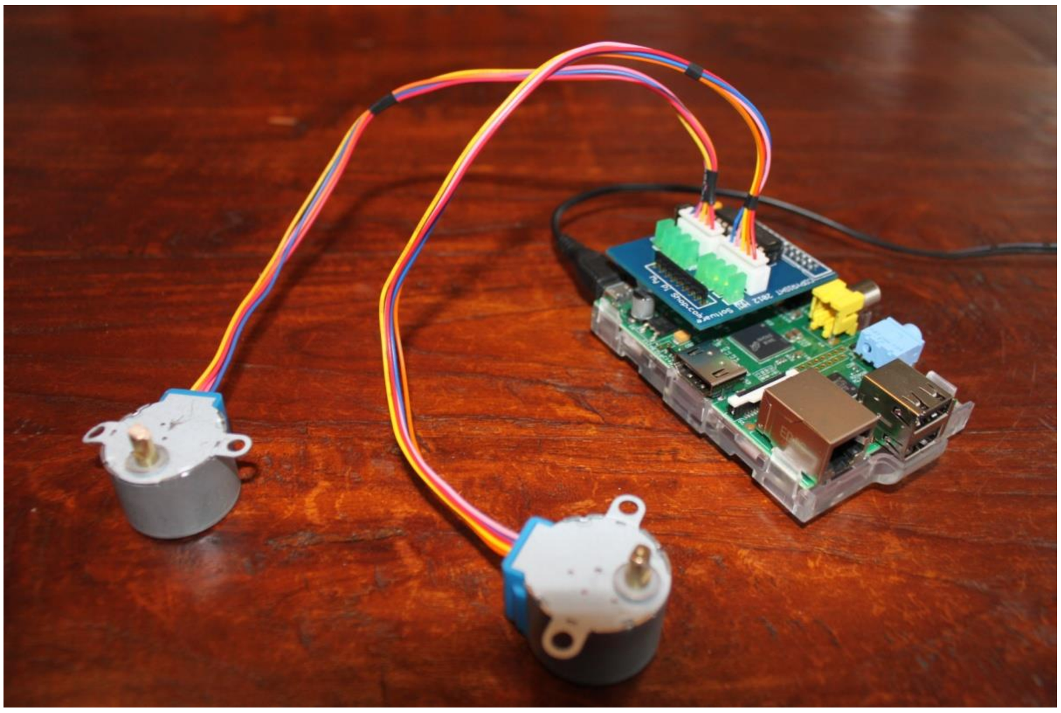
Raspberry PI Stepper Motor

Constructors Manual



Bob Rathbone Computer Consultancy

www.bobrathbone.com

25th of May 2023

Contents

Figures

Tables

Introduction

This project has been designed to help students or hobbyists get started with driving stepper motors on the Raspberry PI. It covers two types of stepper motor namely unipolar and bipolar. The difference between these two is explained in the next section. The principal hardware required to run stepper motors on a Raspberry PI consists of the following components:

- A Raspberry PI computer (all models)
- A single or dual stepper motor driver board
- As alternative to the above an I2C interface can be used
- Object orientated Python3 driver code

Either

- One or two x 5-Wire "28BYJ-48" stepper Motor (bipolar motor) with ULN2803A driver

Or

- A 12-volt #324 (Nema17) high torque stepper motor (unipolar motor) with H-Driver

Raspberry PI computer

The **Raspberry Pi** is a credit-card-sized single-board computer developed in the United Kingdom by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.

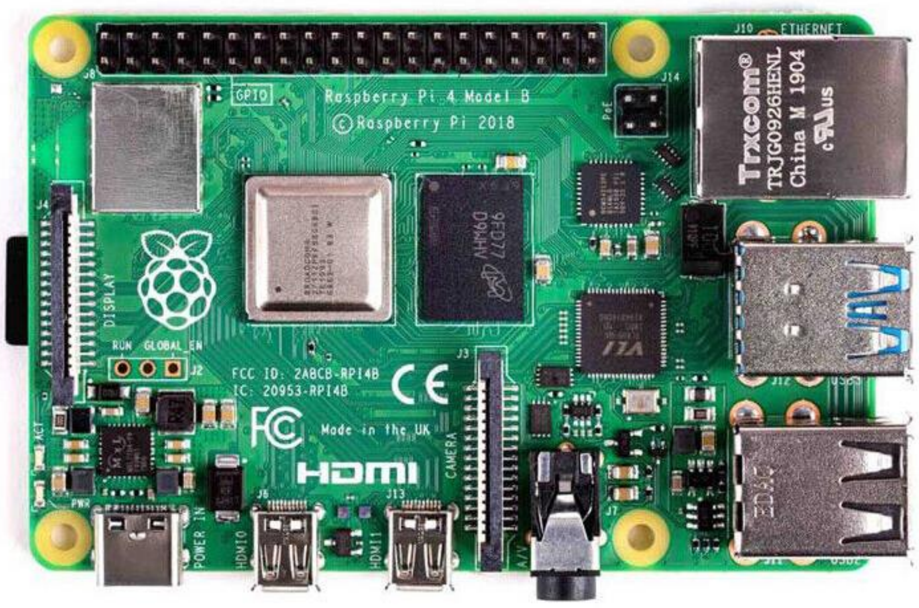


Figure 1 Raspberry PI Computer

More information on the Raspberry PI computer may be found here:
http://en.wikipedia.org/wiki/Raspberry_Pi

If you are new to the Raspberry PI, try the following beginners guide.
http://elinux.org/RPi_Beginners

Stepper Motor Theory

Types of stepper motor

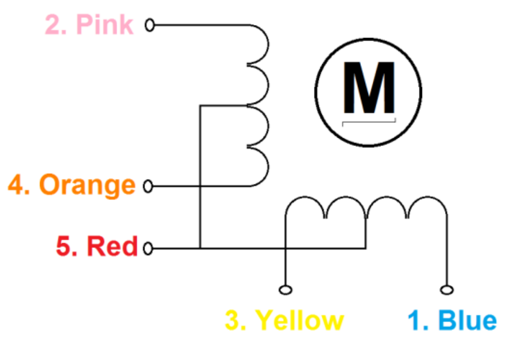
A good place to start is the following Wikipedia Article:
http://en.wikipedia.org/wiki/Stepper_motor

There are two types of stepper motor in popular use. These are:

1. Unipolar stepper motors typically driven using single transistors or Darlington pairs
2. Bipolar motors typically driven using an H-bridge circuit

Unipolar stepper motors

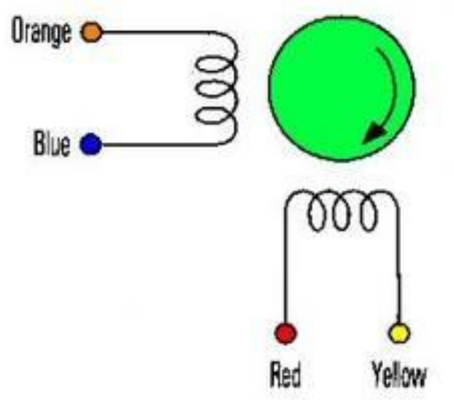
Figure 2 A 5-Wire 28BYJ-48 Stepper Motor Wiring



The 28BYJ-48 stepper motor is a so-called unipolar motor. A unipolar stepper motor has two or more windings, each with centre tap. Each section of windings is switched on for each direction of magnetic field. Since in this arrangement a magnetic pole can be reversed without switching the direction of current, the circuit can be made very simple (e.g., a single transistor) for each winding. In this project the ULN2803A Integrated Circuit is used. This is an eight Darlington pair driver circuit. These motors can normally be driven from 5-volt logic circuits.

Bipolar stepper motors

Figure 3 A 4-Wire Bipolar Stepper Motor Wiring

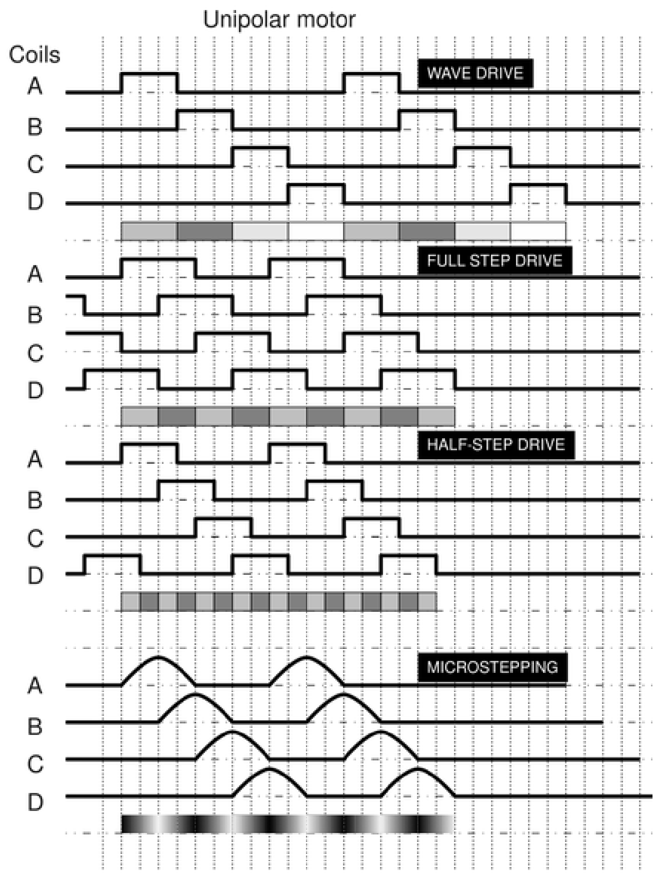


Bipolar stepper motors have a single winding per phase. The current in each winding needs to be reversed in order to reverse the magnetic pole, so the driving circuit is more complicated, typically with an **H-bridge** arrangement, however there are several off-the-shelf driver chips available to make this a simple affair. This project is using the **A4988 H-bridge** circuit driver board. There are two leads per phase, none are common. Bipolar motors are more efficient than unipolar motors as both phases are used at once. They can deliver higher torque and speed than a unipolar motor of the same weight. These motors usually require much higher currents than can be obtained from 5V logic (typically 10 times greater) and will require 8 to 12 volts or higher.

Unipolar driver

There are a number of ways to drive a unipolar stepper motor as shown below

Figure 4 Unipolar stepper motor drive methods



Wave drive

In this drive method only a single phase is activated at a time. It is the fastest drive method but is rarely used.

Full step drive

This is the usual method for full step driving the motor. Two phases are always on so the motor will provide its maximum rated torque.

Half stepping

When half stepping, the drive alternates between two phases on and a single phase on. This increases the angular resolution, but the motor also has less torque.

Microstepping

Here the windings are driven with sinusoidal AC waveform to give smoother operation. This requires different hardware and isn’t used in this project.

Wiring and construction

Raspberry Pi 40-pin GPIO header

The following shows the pin outs for the GPIO for models 2B, 3B and 4B
See: http://elinux.org/RPi_Low-level_peripherals. For more details.

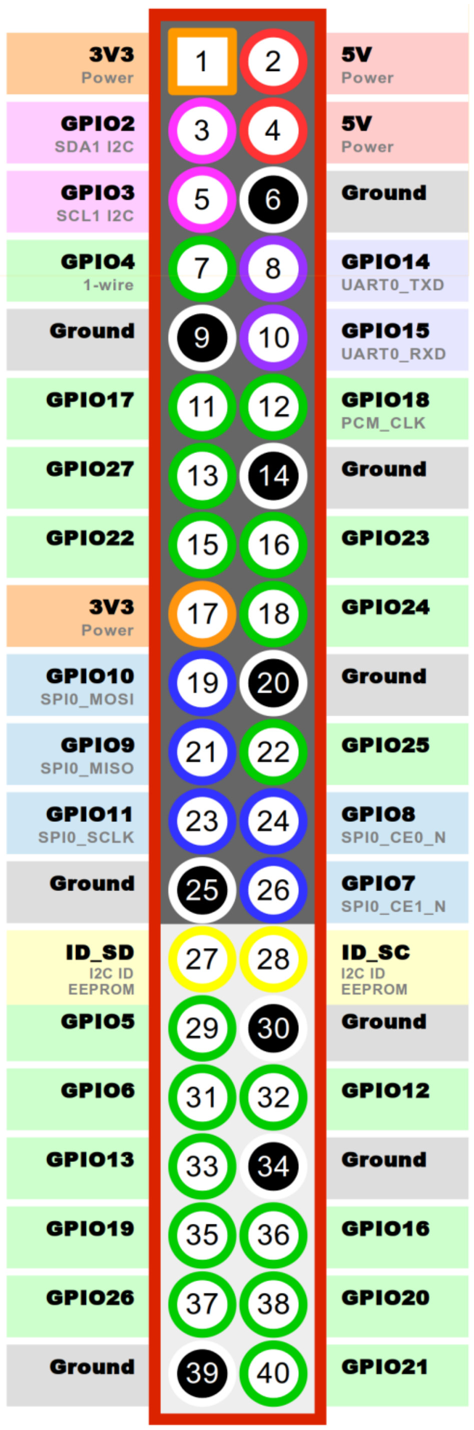


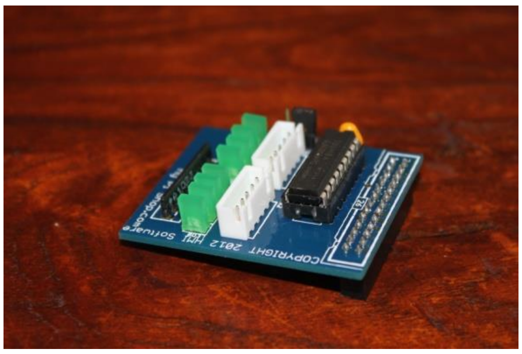
Figure 5 GPIO Numbers

The above diagram shows the GPIO 40 pin header viewed from above.

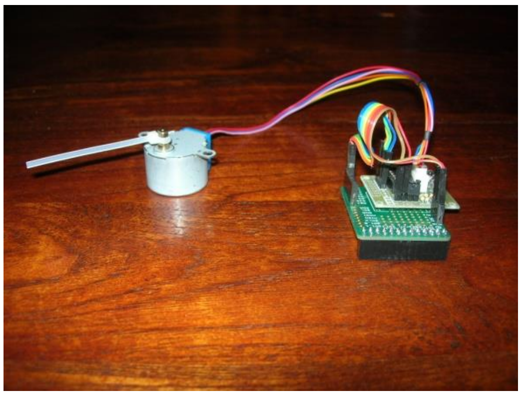
Unipolar Motor driver boards

ULN2803A Darlington pair driver boards

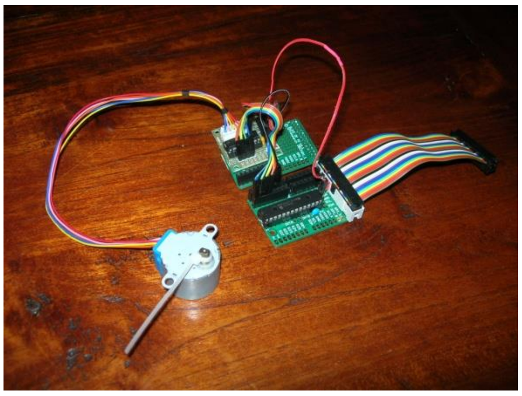
A number of stepper motor driver boards are available.



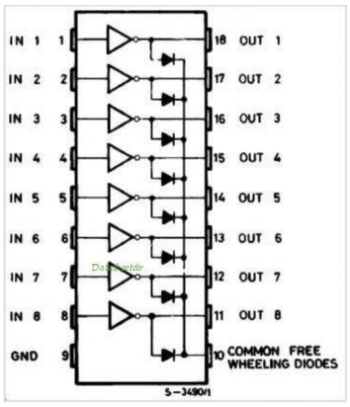
This board is available from ModMyPi and uses the ULN2803A Eight Darlington outputs Driver Chip. It can drive two stepper motors. This board can drive unipolar devices up to 50 volts.



This is an example of a single stepper motor driver board. It uses four outputs of a ULN2003A Seven Darlington outputs Driver Chip. It is piggy-backed on top of a slice of ModPi prototype board using a glue gun. This allows it to be plugged into the GPIO header of the Raspberry Pi.



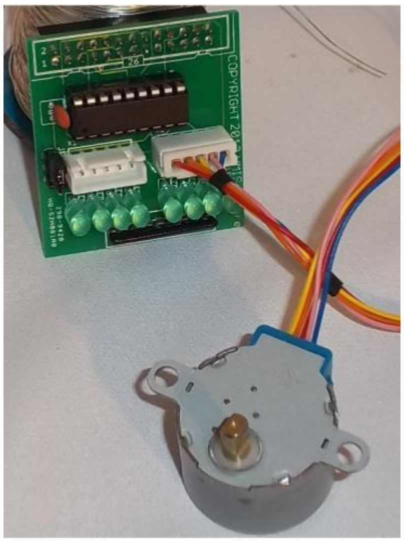
This example shows the I2C interface using a 16 port MCP23017 I/O expander available from ModMyPi or Ciseco. It is connecting to the above single stepper motor driver board. Sixteen I/O ports mean that this board can drive up to four motors using only two pins (I2C) on the Raspberry PI. Since I2C can support up to eight devices many more motors can be driven. Unfortunately, this hardware appears no longer to be available but may be in the future.



The diagram on the left shows the ULN2803A Eight Darlington outputs Driver Chip. This chip can drive two bipolar stepper motors (four outputs are used for each motor.

Construction

Figure 6 Unipolar motor and driver board



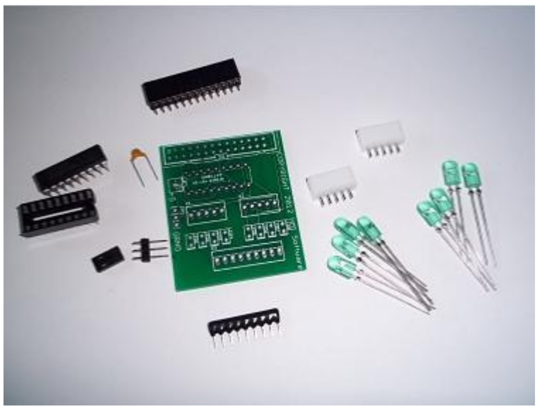
The unipolar stepper motor board uses 8 I/O pins to drive up to two stepper motors.

The jumper shipped with the board allows the stepper motor to use the +5V from the Raspberry PI. If you want to use a different stepper you can remove the jumper and supply up to 12 volts to the centre pin and connect ground to the pin that had no connection.

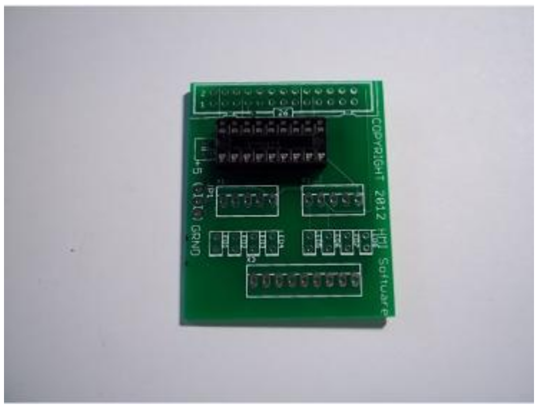
The left white five pin connector is for the first motor and the right connector is for the second motor.

Table 1 GPIO interface wiring

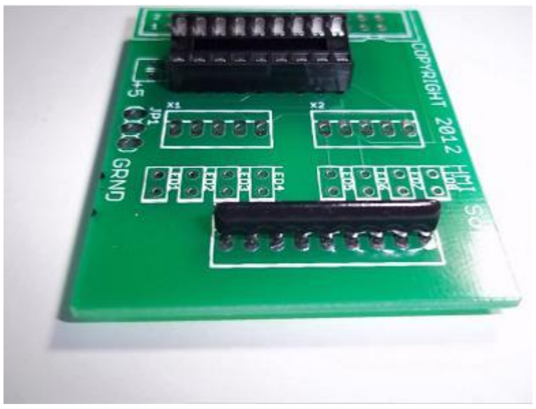
GPIO pin	Connector	Description
17	1	Motor 1 output 1
18	1	Motor 1 output 2
27	1	Motor 1 output 3
22	1	Motor 1 output 4
23	2	Motor 2 output 1
24	2	Motor 2 output 2
25	2	Motor 2 output 3
4	2	Motor 2 output 4

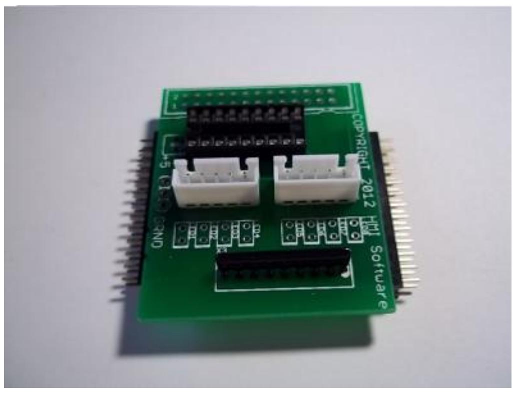


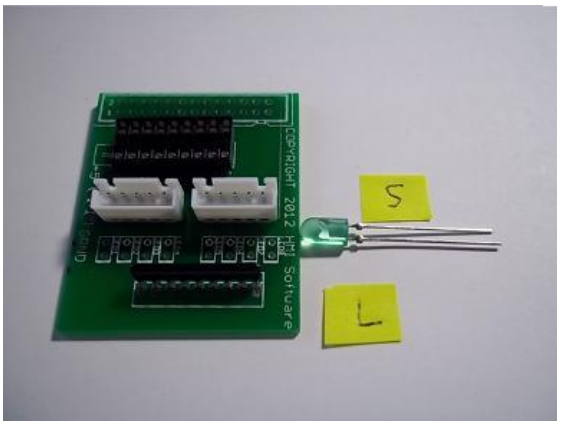
The driver board comes as a kit. Not shown is a small stick-on plastic pad to prevent the card shorting on the Raspberry PI board.



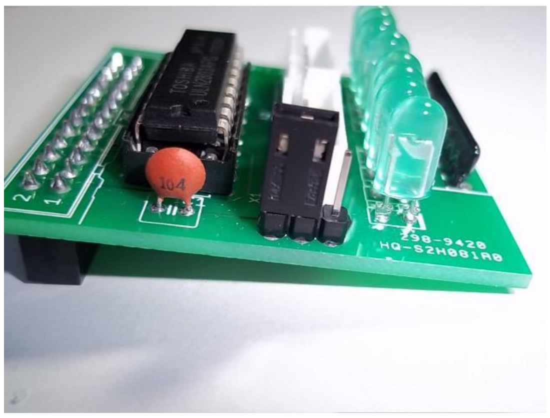
Install and solder IC socket first. Orientate the notch at one end of the socket to the left side of the board.


Install resistor pack. Be sure to orient it as shown. I.E. the text on the resistor pack cannot be seen from this side.


Install white connectors. Be sure to orient them as shown.



Install LED's. Be sure to orient them as shown. Long Leg towards resistor pack.


Insert and solder the capacitor. Solder the 26-pin female socket to the underside of the board as shown above. Insert the ULN2803A Motor Driver Chip with the notch towards the capacitor. Insert and solder the power supply pins and insert the jumper as shown. Finally stick the plastic pad underneath the resistor block in such a way that it rests on the power supply capacitor on the Raspberry PI board.

Bipolar stepper motor A4998 driver

This project uses the A4988 H-circuit driver board to drive the Nema17 stepper motor.

Note: Note that the motor requires a between 8 and 12 volts + connected to VMOT and GND. Take care not to accidentally connect it to VDD as this will destroy the Raspberry Pi.

Figure 7 A4988 driver circuit connection to Raspberry Pi

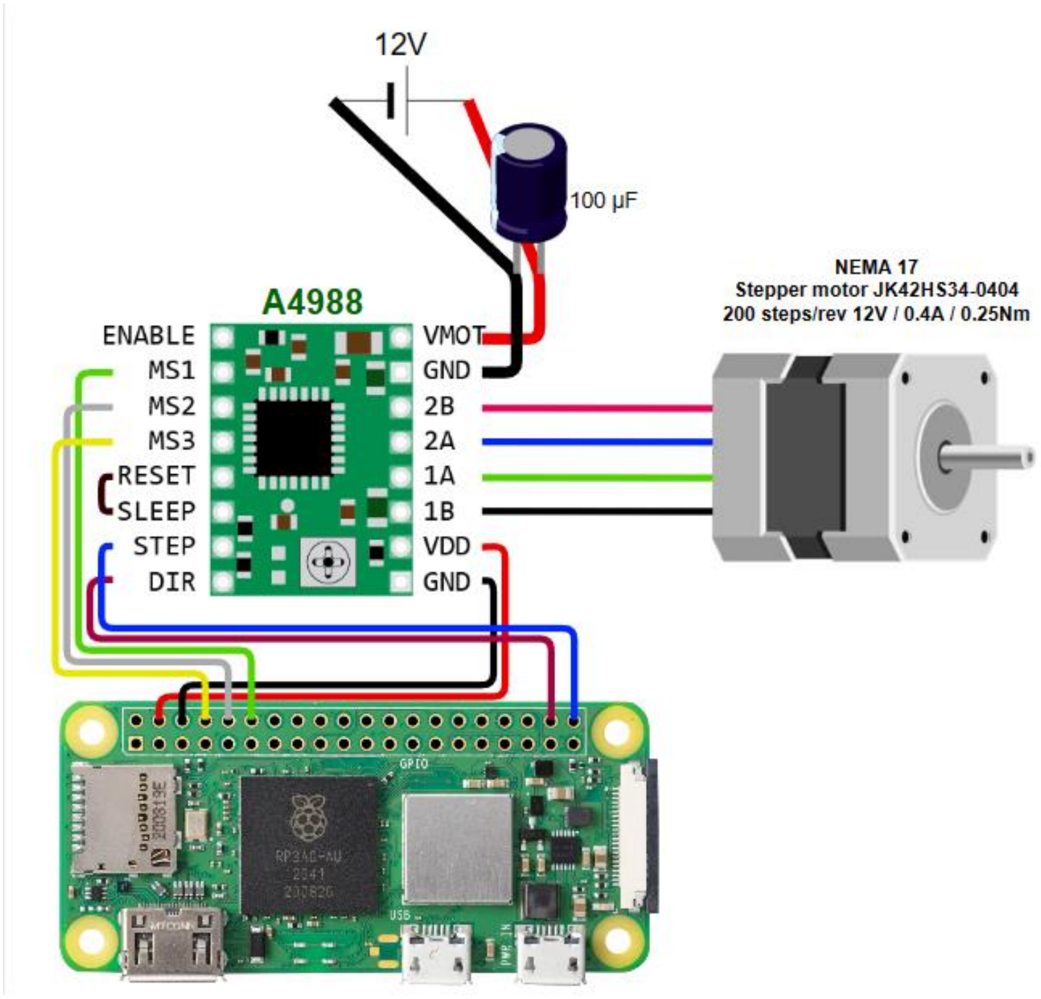


Table 2 A4988 to Raspberry Pi 40 pin header wiring

GPIO/SUPPLY	Physical pin	A4988 Signal	Description
20	38	STEP	Step motor control
21	40	DIR	Direction control
14	8	MS1	Driver signal 1
15	10	MS2	Driver Signal 2
18	12	MS3	Driver signal 3
5V	4	VDD	5V supply
GND	6	GND	Ground 0V
n/a	n/a	! SLEEP	Wire to RESET
n/a	n/a	! RESET	Wire to SLEEP
n/a	n/a	! ENABLE	Not connected, should be LOW
n/a	n/a	VMOT	Motor Voltage 8-12 Volts +
n/a	n/a	GND	Motor voltage GND (0v)

Early versions of the Raspberry Pi only had a 26-pin header. Below is the original wiring for the GPIO inputs for Raspberry Pi's with a 26-pin header.

Note: The original design used ENABLE on GPIO 25 (physical pin 22), which the 40-pin version doesn't. However, you can now leave GPIO 25 disconnected. The ENABLE pin will be held low by the A48AA circuitry. LOW is enabled and HIGH is disabled.

Table 3 A4988 to Raspberry Pi 26 pin header wiring

GPIO/SUPPLY	Physical pin	A4988 Signal	Description
24	18	STEP	Step motor control
4	7	DIR	Direction control
25	22	ENABLE	Enable motor - Can be left disconnected
23	16	MS1	Driver signal 1
22	15	MS2	Driver Signal 2
27	13	MS3	Driver signal 3

The bipolar A4988 H-Bridge driver board

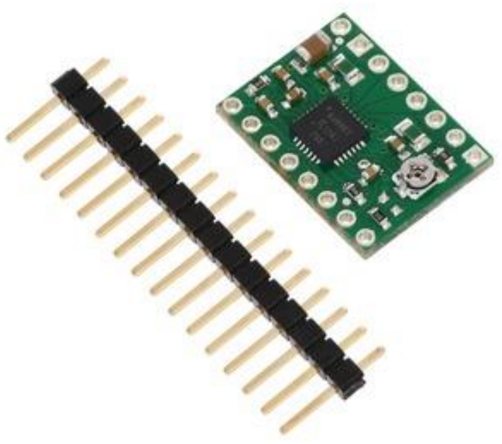


Figure 8 A4988 H-bridge driver board kit

The A4988 H-Bridge driver board comes as a simple kit. Break the in-line connector in half and solder the short pins into the board. The long pins can then connect directly into a breadboard or can be connected via matching female connectors into an interface PCB such as the ModMyPi Humble PI.

You can solder the pins as shown or turn the board upside down and solder the pins in from the other side so that you can read the pin names which is much more convenient.

A4988 H-Bridge circuit

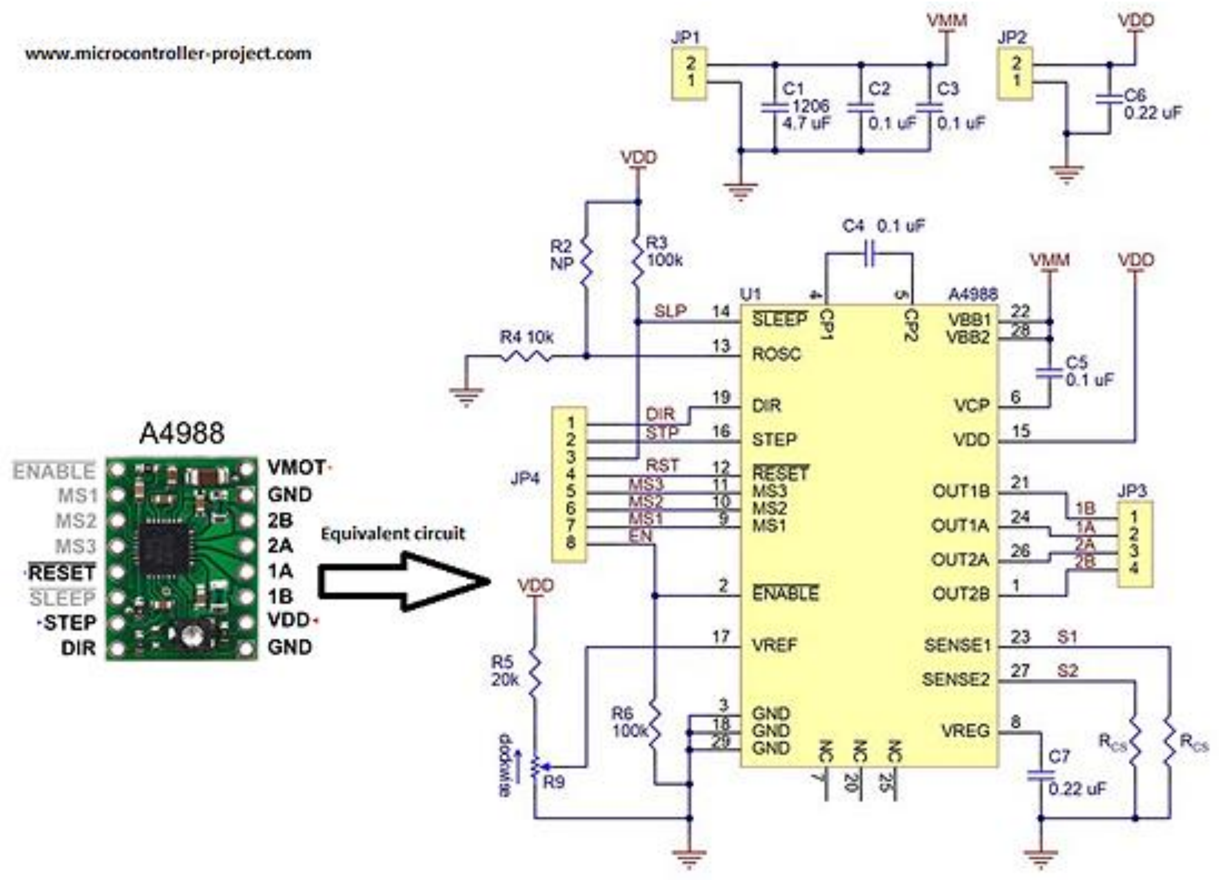


Figure 9 A4988 Circuit Diagram – Courtesy MicroController-project

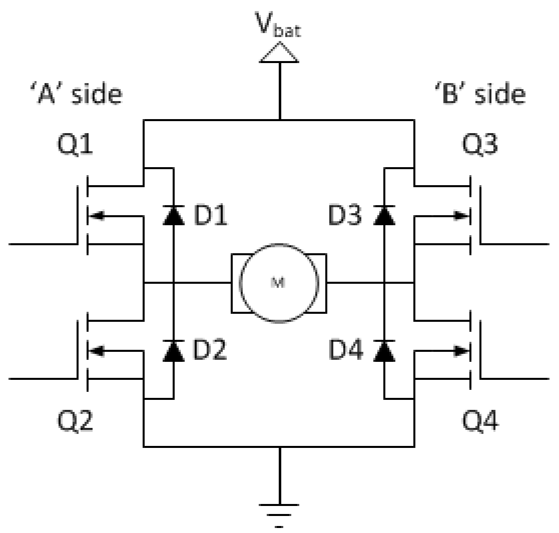


Figure 10 H-Bridge circuit

In general, an H-bridge is a rather simple circuit, containing four switching elements, with the load at the centre, in an H-like configuration.

The switching elements (Q1..Q4) are usually bi-polar or FET transistors, in some high-voltage applications IGBTs. Integrated solutions also exist but whether the switching elements are integrated with their control circuits or not is not relevant for the most part for this discussion. The diodes (D1..D4) are called catch diodes and are usually of a Schottky type.

Source code and usage

The code for driving the motor comes as a number of separate source files.
To extract software on the Raspberry Pi first download with **wget** and then extract it with **tar.**

```
$ mkdir stepper
$ cd stepper
$ wget https://bobrathbone.com/raspberrypi/packages/pi_stepper_motor.tar.gz
$ tar -xvf pi_stepper_motor.tar.gz
```

unipolar_class.py

This is the actual code that drives the 28BYJ-48 motor using the GPIO pins. It is called by various other 28BYJ-48 driver programs. To use the class in a program first import it and define the motor(s). In the following we define two motors and the GPIO pins they will be using.

```
from unipolar_class import Motor
motora = Motor(17,18,27,22)
motorb = Motor(4,25,24,23)
```

Before a motor can be used it must be initialised. This sets up the GPIO pins.

```
motora.init()
```

To turn the motor one revolution clockwise:

```
motora.turn(1*Motor.REVOLUTION, Motor.CLOCKWISE)
```

To turn the motor two revolutions anti-clockwise:

```
motora.turn(2*Motor.REVOLUTION, Motor.ANTICLOCKWISE)
```

To turn the motor two steps anti-clockwise:

```
motora.turn(2, Motor.ANTICLOCKWISE)
```

The above turns the shaft 0.7 degrees per step (360/512 = 0.703125 Degrees)
The motor has 512 positions. To turn the motor to a particular position (200 in this case):

```
motora.goto(200)
```

To lock the motor in its current position:

```
motora.lock()
```

To stop an already turning motor:

```
motora.interrupt()
```

To set the type of stepping (See *Stepper Motor Theory* on page 5) use one of the following calls.

```
motora.setFullStepDrive()
motora.setHalfStepDrive()
motora.setWaveDrive()
```

The default is *Full Step Drive*. It isn't necessary to set this as it is the default.

test_unipolar_class.py

This contains simple examples of driving two motors using the dual motor driver board.

motor_i2c_class.py

This class does the same as the **motor_class.py** code but uses the i2C interface.

test_motor_ic2_class.py

This class does the same as the **test_motor_class.py** code but uses the i2C interface. However, the motor definitions are different. The MCP23017 I/O expander chip has two banks of eight I/O ports making sixteen in all which allows up to four motors to be driven per MCP23017 I/O expander.

```
address = 0x20 # I2C address of MCP23017
motora = Motor(address,Motor.MOTOR_A)
motorb = Motor(address,Motor.MOTOR_B)
motorc = Motor(address,Motor.MOTOR_C)
motord = Motor(address,Motor.MOTOR_D)
```

The address parameter is normally Hex 0x20 for the MCP23017 I/O expander chip. See *The MCP23017 chip* on page 19 and *Installing the I2C libraries* on page 17 for more information.

motord.py

The **motord.py** program is a more complex example of driving two motors concurrently. It runs as a system daemon. Each motor is handled by a separate (forked) process. This allows the motors to be turned at the same time.
Just invoking the program displays its usage.

```
$ sudo ./motord.py
usage: ./motord.py start|stop|restart|status|version
```

To start and stop the motor daemon, use the following code.

```
$ sudo ./motord.py start
$ sudo ./motord.py stop
```

Note: The current motor command will be always completed when the stop command is issued.
If the **motord** daemon is running then issuing the status command will display its PID.

```
$ sudo ./motord.py status
Motor daemon running pid 2813
```

The **pid** will be different each time the **motord** program is run.
The version command shows the current version of the software.

```
$ sudo ./motord.py version
Version 1.0
```

motor_daemon.py

This is the code to create the daemon process and to start and stop it. It is used by the **motord.py** program only.

The Log class

The *log_class.py* routine provides logging of events to **/var/log/motor.log** file. It is used by the **motord.py** program only. It logs to **/var/log/motor.log.** The log level needs to be set up in **/var/lib/motor/loglevel** file and should contain one of the following:

INFO, WARNING, ERROR or DEBUG

The bipolar class

This is the low-level driver for the NEMA17 high torque stepper motor.
Any high-level program such as **test_nema17.py** must first import this class as shown below:

```
from bipolar_class import Motor
```

This makes use of six GPIOs to drive the A4988 H-Bridge circuit. They are defined the following, for example **GPIO 20** defines the **step** signal. Below are the definitions for Raspberry Pi's with a 40-pin header.

```
# 40 pin header for newer Raspberry Pi's
step = 20
direction = 21
enable = 25     # Optional - can be left unconnected
ms1 = 14
ms2 = 15
ms3 = 18
```

The test program is **test_nema17.py**.

There are also definitions for Raspberry Pi's with a 26-pin header or interface boards with 26 pins.

```
# 26 pin header for older Raspberry Pi's
step = 24
direction = 4
enable = 25
ms1 = 23
ms2 = 22
ms3 = 27
```

The test program for a 26-pin header is **test_26_nema17.py**.

The test_ema17 test program

This is the top-level program to drive the NEMA17 stepper motor. It uses the bipolar_class.py driver. It uses the Raspberry Pi 40 pin header.

The test_26_ema17 test program

This is the same as the above program but uses the Raspberry Pi 26 pin header.

Other files

single_motor.py	Test a single 28BJY48 unipolar motor
test_position.py	Positional tests based upon number of steps. One revolution = 256 steps

Software installation

This procedure assumes that the Raspberry PI is installed with Debian Wheezy and with a working Internet Connection. There are several steps to carry out to install the software.

- Download and un-tar the software from the Bob Rathbone web site
- Optionally install I2C library (If using the I2C programs)

Software download

The software is contained in a compressed tar file called *pi_stepper_motor.tar.gz*. This can be downloaded from the following location.

https://bobrathbone.com/raspberrypi/stepper_motor.html

Either download it to the PC and copy it across to the Raspberry PI or use the *wget* facility if there is an Internet connection on the Raspberry PI.

Create a directory called **/home/pi/stepper**. Copy the *pi_stepper_motor.tar.gz* to the **/home/pi/stepper** directory or use **wget** to download it.

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/pi_stepper_motor.tar.gz
```

Un-tar the file with the following command:

```
$ tar -xvf pi_stepper_motor.tar.gz
```

This will unzip the following files and directory:
test_unipolar_class.py, unipolar_class.py, single_motor.py, test_position.py, motord.py, motor_daemon.py, log_class.py, bipolar_class.py, test_nema17.py, test_26_nema17.py, test_motor_i2c_class.py motor_i2c_class.py, README

GitHub

The source for this project is also available from GitHub at:
https://github.com/bobrathbone/pistepper

Installing the I2C libraries

If you are using the **motor_i2c_class.py** and **test_motor_i2c_class.py** programs it is necessary to install the I2C libraries. If not then skip this section. As the hardware required to run these programs appears to be no longer available, they have not been converted to Python 3 but are included in this release if you have the old hardware.

Edit **/etc/modules** file and add the following lines to the end of the file. Then save and reboot to enable the hardware I2C driver.

```
i2c-bcm2708
i2c-dev
```

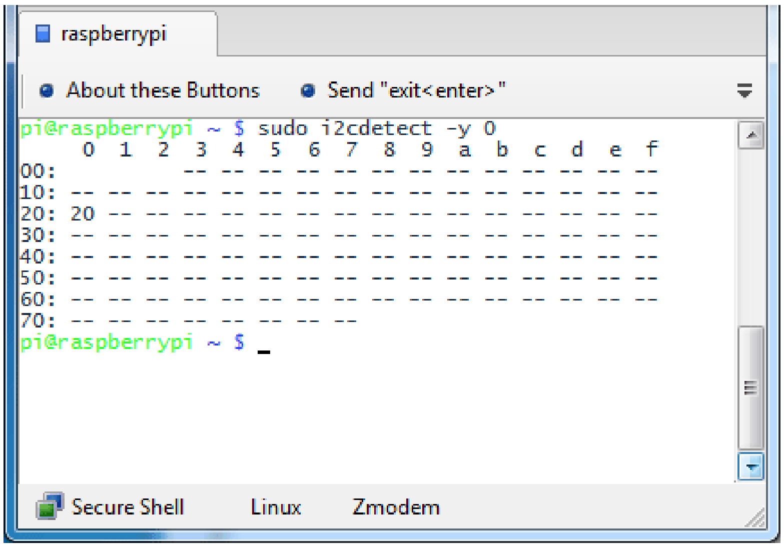
Enter the following commands to add SMBus support (which includes I2C) to Python:

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

The **i2c-tools** package isn't strictly required, but it's a useful package since you can use it to scan for any I2C or SMBus devices connected to the Raspberry. If you know something is connected, but you don't know it's 7-bit I2C address, this library has a great little tool to help you find it:

```
sudo i2cdetect -y 0 (if you are using a version 1 Raspberry Pi)
sudo i2cdetect -y 1 (if you are using a version 2 Raspberry Pi)
```

This will search **/dev/i2c-0** or **/dev/i2c-1** for all address, and if the ModMyPi I2C interface is correctly connected, it should show up at **0x20.**



```
pi@raspberrypi ~ $ sudo i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi ~ $ _
```

Once both of these packages have been installed, you have everything you need to get started accessing I2C and SMBus devices in Python.

Configure motord.py program log rotation

If you will not be using the **radiod.py** daemon then skip this section.

The Radio program logs to a file called **/var/log/motor.log**. This can eventually fill the SD card. Create a file called **/etc/logrotate.d/motor** with the following lines:

```
    /var/log/motor.log {
        weekly
        missingok
        rotate 7
        compress
        notifempty
        copytruncate
        create 600
    }
```

This will rotate the log files every week so prevent the SD card from eventually filling up.

The MCP23017 expander board

If you are connecting the stepper motor using the I2C interface then you will need an I/O expander board.

There are a number of expander boards available as shown in the following figure. For this project we are using the one shown on the right from Ciseco.

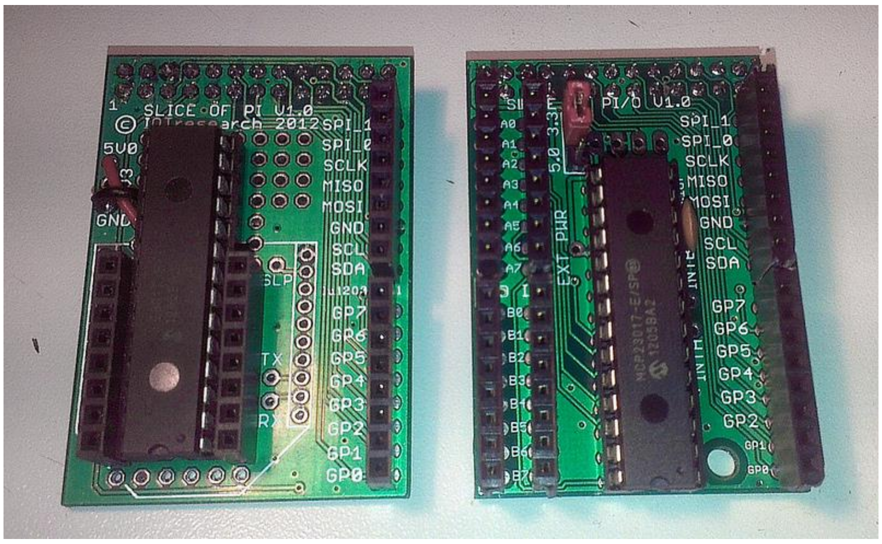


Figure 11 MCP23017 expander boards

Please note in this picture the B0 to B7 outputs are labelled the wrong way round. B7 at the bottom should be B0 and so on.

The MCP23017 chip

The following diagram shows the pin outs for the MCP23017

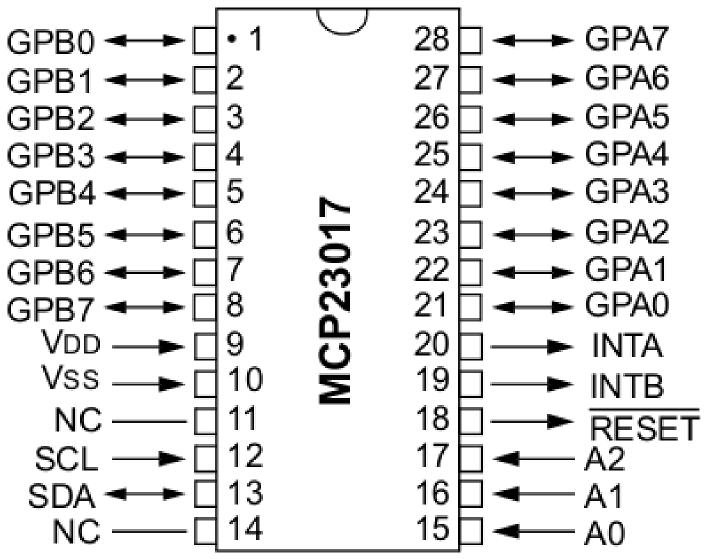


Figure 12 The MCP23017 chip

There are two output banks A and B of 8 pins each. The **I²C** interface consists of a data (SDA) and clock (SCL). The chip has up to eight addresses by biasing the A0, A1 and A2 lines. The full specification for the MCP23017 chip can be found at:
http://ww1.microchip.com/downloads/en/devicedoc/21952b.pdf

MCP2317 and Ciseco Board signals

The MCP2317 chip has 16 input/output pins as shown in the following table. This shows the hex, decimal or binary values that have to be written to the MCP2317 chip enable the outputs. The banks A and B are addressed by 0x12 and 0x13 respectively (See example program listings in appendix A).

I/O	MCP23017	Pin	Ciseco Board	Bank	Hex	Decimal	Binary
1	GPA0	1	A0	0x12	0x01	1	00000001
2	GPA1	2	A1	0x12	0x02	2	00000010
3	GPA2	3	A2	0x12	0x04	4	00000100
4	GPA3	4	A3	0x12	0x08	8	00001000
5	GPA4	5	A4	0x12	0x10	16	00010000
6	GPA5	6	A5	0x12	0x20	32	00100000
7	GPA6	7	A6	0x12	0x40	64	01000000
8	GPA7	8	A7	0x12	0x80	128	10000000
9	GPB0	21	B0 (B7)	0x13	0x01	1	00000001
10	GPB1	22	B1 (B6)	0x13	0x02	2	00000010
11	GPB2	23	B2 (B5)	0x13	0x04	4	00000100
12	GPB3	24	B3 (B4)	0x13	0x08	8	00001000
13	GPB4	25	B4 (B3)	0x13	0x10	16	00010000
14	GPB5	26	B5 (B2)	0x13	0x20	32	00100000
15	GPB6	27	B6 (B1)	0x13	0x40	64	01000000
16	GPB7	28	B7 (B0)	0x13	0x80	128	10000000

Note: The first batch of Ciseco expander boards have B0-B7 labelled the wrong way round. The numbers in brackets are the incorrect labelling. Watch out for this. Later batches of this board should be correct.

If for example you wish to enable I/O 9 (GPB0) and 10 (GPB1) together you must enable bank 1 and then either add the decimal values together. 1 + 2 = 3 = Hex 0x3 = binary 00000011.

Appendix A Code Listings

All code can be downloaded from the following URL:

https://bobrathbone.com/raspberrypi/packages/pi_stepper_motor.tar.gz

The following table lists all of the available software and its function.

Table 4 Source files

File name	Driver Class	Type	Description
test_unipolar_class.py	unipolar_class.py	Unipolar	28BYJ-48 stepper dual motor driver
single_motor.py	unipolar_class.py	Unipolar	28BYJ-48 stepper single motor
test_motor_i2c_class.py	test_motor_i2c_class.py	Unipolar	28BYJ-48 stepper motor I2C driver
test_position.py	unipolar_class.py	Unipolar	28BYJ-48 test position setting
motord.py	motor_daemon.py	Unipolar	28BYJ-48 background daemon
log_class.py	n/a	n/a	Logging class for motord.py
test_nema17.py	bipolar_class.py	Bipolar	Nema17 stepper motor driver
test_26_ema17.py	bipolar_class.py	Bipolar	Nema17 driver 26-pin header
test_motor_i2c_class.py	motor_i2c_class.py	Unipolar	I2C 28BYJ-48 stepper dual motor driver (1)

Note 1: The hardware for the I2C interface appears no longer to be available but the I2C programs are included in case you have this old hardware. The I2C programs written in Python 2 as they cannot be tested with Python 3.

Appendix B – Specifications

Appendix B.1 - 28BYJ-48 – 5V Stepper Motor

• Operating Voltage	5VDC
• Operating Current	240mA (typical)
• Number of phases	4
• Gear Reduction Ratio	64:1
• Step Angle	5.625°/64
• Frequency	100Hz
• In-traction Torque	>34.3mN.m(120Hz)
• Self-positioning Torque	>34.3mN.m
• Friction torque	600-1200 gf.cm
• Pull in torque	300 gf.cm

The 28BYJ-48 data sheet can be found at:
https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf

Appendix B.2 – Nema17 2-phase Stepper Motor

- Rated Voltage: 12V DC
- Current: 1.2A at 4V
- Step Angle: 1.8 deg.
- No. of Phases: 4
- Motor Length: 1.54 inches
- 4-wire, 8 inch lead
- 200 steps per revolution, 1.8 degrees
- Operating Temperature: -10 to 40 °C
- Unipolar Holding Torque: 22.2 oz-in

The Nema17 data sheet can be found at:
https://datasheetspdf.com/pdf-file/1260602/Schneider/NEMA17/1

Appendix C Licences

The software and documentation for this project is released under the GNU General Public Licence.

The GNU General Public License (GNU GPL or GPL) is the most widely used free software license, which guarantees end users (individuals, organizations, companies) the freedoms to use, study, share (copy), and modify the software. Software that ensures that these rights are retained is called free software. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU project.

The GPL grants the recipients of a computer program the rights of the Free Software Definition and uses *copyleft* to ensure the freedoms are preserved whenever the work is distributed, even when the work is changed or added to. The GPL is a *copyleft* license, which means that derived works can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses are the standard examples. GPL was the first *copyleft* license for general use. This means that you may modify and distribute the software and documentation subject to the conditions of the licences.

See http://www.gnu.org/licenses for further information on the GNU General Public License.

The licences for the source and documentation for this project are:

GNU General Public License.	See http://www.gnu.org/licenses/gpl.html
GNU AFFERO General Public License.	See http://www.gnu.org/licenses/agpl.html
GNU Free Documentation License.	See http://www.gnu.org/licenses/fdl.html

Acknowledgements

Thanks to the numerous Raspberry PI contributors who have placed articles about driving stepper motors using the Raspberry PI on their websites and blogs for the benefit of the community.

The code for the 28BYJ48 stepper motor is based upon original code from PiHut.

Some diagrams came from Aleksas Pielikis at https://github.com/aleksas/zero-stepper

Glossary

GND	Ground (0 Volts)
GPIO	General Purpose IO (On the Raspberry PI)
IC	Integrated Circuit
NEMA	The US National Electrical Manufacturers Association
PID	Process Identification Number
VDD	Voltage Drain Drain (In this project +5 Volts to the **A4988 H-bridge** circuit driver board)
VMOT	Voltage Motor (In this project +8 to 12 Volt power to the **Nema17** stepper motor)