http://www.raspberrypi-spy.co.uk/2012/07/16x2-lcd-module-control-using-python/

Once you’ve played with LEDs, switches and stepper motors the next natural step is 16×2 alphanumeric LCD modules. These modules are cheap (less than $10) and easy to interface to the Raspberry Pi. They have 16 connections but you only need to use 6 GPIO pins on your

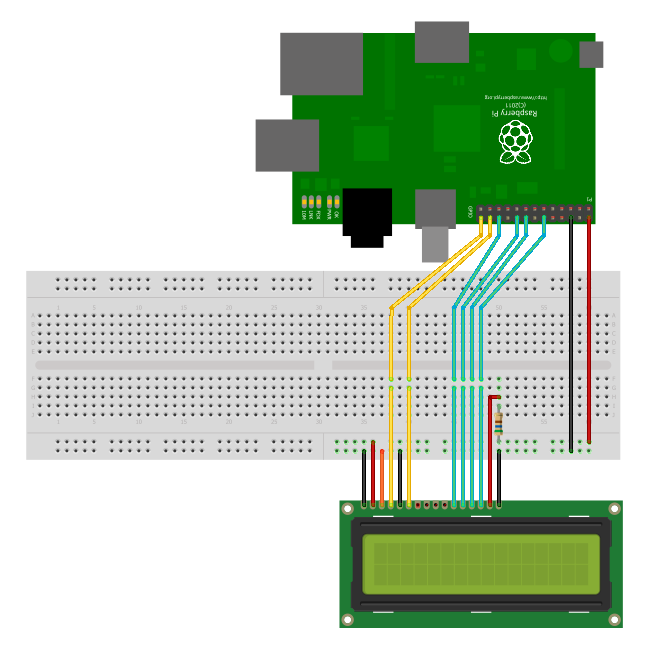
Here is how I wired up my LCD :

|  |  |  |  |
| --- | --- | --- | --- |
| LCD Pin | Function | Pi Function | Pi Pin |
| 01 | GND | GND | P1-06 |
| 02 | +5V | +5V | P1-02 |
| 03 | Contrast | GND | P1-06 |
| 04 | RS | GPIO7 | P1-26 |
| 05 | RW | GND | P1-06 |
| 06 | E | GPIO8 | P1-24 |
| 07 | Data 0 |  |  |
| 08 | Data 1 |  |  |
| 09 | Data 2 |  |  |
| 10 | Data 3 |  |  |
| 11 | Data 4 | GPIO25 | P1-22 |
| 12 | Data 5 | GPIO24 | P1-18 |
| 13 | Data 6 | GPIO23 | P1-16 |
| 14 | Data 7 | GPIO18 | P1-12 |
| 15 | +5V via 560ohm |  |  |
| 16 | GND |  | P1-06 |

**NOTE :** The RW pin allows the device to be be put into read or write mode. I wanted to send data to the device but did not want it to send data to the Pi so I tied this pin to ground. The Pi can not tolerate 5V inputs on its GPIO header. Tying RW to ground makes sure the device does not attempt to pull the data lines to 5V which would damage the Pi.

In order to control the contrast you can adjust the voltage presented to Pin 3. This must be between 0 and 5V. I tied this pin to ground.

Pin 15 provides 5V to the backlight LED. It wasn’t clear on my device if this could be connected direct to 5V so I played safe and placed a 560ohm resistor in line with this pin.



**Wiring Checks**

Here are some sanity checks before you power up your circuit for the first time :

* Pin 1 (GND), 3 (Contrast), 5 (RW) and 16 (LED -) ( should be tied to ground.
* Pin 2 should be tied to 5V. Pin 15 should have a resistor inline to 5V to protect the backlight.
* Pin 7-10 are unconnected
* Pin 11-14 are connected to GPIO pins on the Pi

**Python**

You can control a HD44780 style display using any programming environment you like but my weapon of choice is Python. I use the [RPi.GPIO library](http://www.raspberrypi-spy.co.uk/2012/07/install-rpi-gpio-library-in-raspbian/" \o "Install RPi.GPIO Library In Raspbian) to provide access to the GPIO.

Here is my code :

|  |  |
| --- | --- |
| 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  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180 | #!/usr/bin/python  #--------------------------------------  #    \_\_\_  \_\_\_  \_ \_\_\_\_  #   / \_ \/ \_ \(\_) \_\_/\_\_  \_\_ \_\_  #  / , \_/ \_\_\_/ /\ \/ \_ \/ // /  # /\_/|\_/\_/  /\_/\_\_\_/ .\_\_/\\_, /  #                /\_/   /\_\_\_/  #  #  lcd\_16x2.py  #  16x2 LCD Test Script  #  # Author : Matt Hawkins  # Date   : 06/04/2015  #  # http://www.raspberrypi-spy.co.uk/  #  #--------------------------------------    # The wiring for the LCD is as follows:  # 1 : GND  # 2 : 5V  # 3 : Contrast (0-5V)\*  # 4 : RS (Register Select)  # 5 : R/W (Read Write)       - GROUND THIS PIN  # 6 : Enable or Strobe  # 7 : Data Bit 0             - NOT USED  # 8 : Data Bit 1             - NOT USED  # 9 : Data Bit 2             - NOT USED  # 10: Data Bit 3             - NOT USED  # 11: Data Bit 4  # 12: Data Bit 5  # 13: Data Bit 6  # 14: Data Bit 7  # 15: LCD Backlight +5V\*\*  # 16: LCD Backlight GND    #import  import RPi.GPIO as GPIO  import time    # Define GPIO to LCD mapping  LCD\_RS = 7  LCD\_E  = 8  LCD\_D4 = 25  LCD\_D5 = 24  LCD\_D6 = 23  LCD\_D7 = 18    # Define some device constants  LCD\_WIDTH = 16    # Maximum characters per line  LCD\_CHR = True  LCD\_CMD = False    LCD\_LINE\_1 = 0x80 # LCD RAM address for the 1st line  LCD\_LINE\_2 = 0xC0 # LCD RAM address for the 2nd line    # Timing constants  E\_PULSE = 0.0005  E\_DELAY = 0.0005    def main():    # Main program block    GPIO.setwarnings(False)    GPIO.setmode(GPIO.BCM)       # Use BCM GPIO numbers    GPIO.setup(LCD\_E, GPIO.OUT)  # E    GPIO.setup(LCD\_RS, GPIO.OUT) # RS    GPIO.setup(LCD\_D4, GPIO.OUT) # DB4    GPIO.setup(LCD\_D5, GPIO.OUT) # DB5    GPIO.setup(LCD\_D6, GPIO.OUT) # DB6    GPIO.setup(LCD\_D7, GPIO.OUT) # DB7      # Initialise display    lcd\_init()      while True:        # Send some test      lcd\_string("Rasbperry Pi",LCD\_LINE\_1)      lcd\_string("16x2 LCD Test",LCD\_LINE\_2)        time.sleep(3) # 3 second delay        # Send some text      lcd\_string("1234567890123456",LCD\_LINE\_1)      lcd\_string("abcdefghijklmnop",LCD\_LINE\_2)        time.sleep(3) # 3 second delay        # Send some text      lcd\_string("RaspberryPi-spy",LCD\_LINE\_1)      lcd\_string(".co.uk",LCD\_LINE\_2)        time.sleep(3)        # Send some text      lcd\_string("Follow me on",LCD\_LINE\_1)      lcd\_string("Twitter @RPiSpy",LCD\_LINE\_2)        time.sleep(3)    def lcd\_init():    # Initialise display    lcd\_byte(0x33,LCD\_CMD) # 110011 Initialise    lcd\_byte(0x32,LCD\_CMD) # 110010 Initialise    lcd\_byte(0x06,LCD\_CMD) # 000110 Cursor move direction    lcd\_byte(0x0C,LCD\_CMD) # 001100 Display On,Cursor Off, Blink Off    lcd\_byte(0x28,LCD\_CMD) # 101000 Data length, number of lines, font size    lcd\_byte(0x01,LCD\_CMD) # 000001 Clear display    time.sleep(E\_DELAY)    def lcd\_byte(bits, mode):    # Send byte to data pins    # bits = data    # mode = True  for character    #        False for command      GPIO.output(LCD\_RS, mode) # RS      # High bits    GPIO.output(LCD\_D4, False)    GPIO.output(LCD\_D5, False)    GPIO.output(LCD\_D6, False)    GPIO.output(LCD\_D7, False)    if bits&0x10==0x10:      GPIO.output(LCD\_D4, True)    if bits&0x20==0x20:      GPIO.output(LCD\_D5, True)    if bits&0x40==0x40:      GPIO.output(LCD\_D6, True)    if bits&0x80==0x80:      GPIO.output(LCD\_D7, True)      # Toggle 'Enable' pin    lcd\_toggle\_enable()      # Low bits    GPIO.output(LCD\_D4, False)    GPIO.output(LCD\_D5, False)    GPIO.output(LCD\_D6, False)    GPIO.output(LCD\_D7, False)    if bits&0x01==0x01:      GPIO.output(LCD\_D4, True)    if bits&0x02==0x02:      GPIO.output(LCD\_D5, True)    if bits&0x04==0x04:      GPIO.output(LCD\_D6, True)    if bits&0x08==0x08:      GPIO.output(LCD\_D7, True)      # Toggle 'Enable' pin    lcd\_toggle\_enable()    def lcd\_toggle\_enable():    # Toggle enable    time.sleep(E\_DELAY)    GPIO.output(LCD\_E, True)    time.sleep(E\_PULSE)    GPIO.output(LCD\_E, False)    time.sleep(E\_DELAY)    def lcd\_string(message,line):    # Send string to display      message = message.ljust(LCD\_WIDTH," ")      lcd\_byte(line, LCD\_CMD)      for i in range(LCD\_WIDTH):      lcd\_byte(ord(message[i]),LCD\_CHR)    if \_\_name\_\_ == '\_\_main\_\_':      try:      main()    except KeyboardInterrupt:      pass    finally:      lcd\_byte(0x01, LCD\_CMD)      lcd\_string("Goodbye!",LCD\_LINE\_1)      GPIO.cleanup() |

This script can be downloaded using [this link](https://bitbucket.org/MattHawkinsUK/rpispy-misc/raw/master/python/lcd_16x2.py) or directly to your Pi using the following command :

wget https://bitbucket.org/MattHawkinsUK/rpispy-misc/raw/master/python/lcd\_16x2.py

It can then be run using :

sudo lcd\_16x2.py

If you use this code the only thing you will need to change is the GPIO pin mapping depending on what pins you use on your Pi GPIO header. Here are some photos :