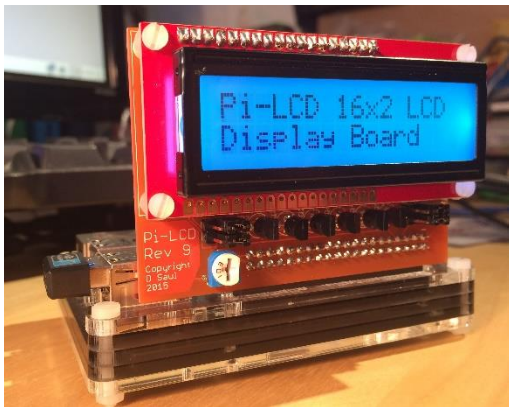
Pi-LCD

The Pi-LCD board allows most common 16x2 LCD modules, including the newer full colour RGB backlit variants to be easily connected to a Raspberry Pi. The board includes 3 buffered drive circuits allowing full control of mono the colour RGB back-lights.

The Pi-LCD can be used in a number of ways, but particularly to speed up development of Pi projects needing an LCD display as it saves you the need to focus on the display element.



1 Introduction – How to use this document

This document contains all should need to build / get working / develop your Pi-LCD. For those with ready built boards, who just want to get something working quickly you can jump to the section 2 'quick start'.

The document is laid out in what you will hopefully find is a logical order starting with build instructions. The build is straightforward but there is the potential to solder parts in the wrong place so please don't dive straight in with your soldering iron without at least glancing at the instructions first.

These instructions assume you are comfortable with basic operation of the Raspberry Pi and the Raspbian operating system. If you are new to the Raspberry Pi you should get yourself familiar with its operation before attempting to connect and operate your Pi-LCD. There are lots of beginner's guides on the internet, the official Raspberry Pi foundation material can be found at www.raspberrypi.org/help/ .

If you are considering the Pi-LCD for educational purposes, lesson plans and additional activity support material is available with the educational Pi-LCD packs. These can also be adapted to suit specific needs if required.

1.1 Do's and don'ts

The Pi-LCD connects directly to the Raspberry Pi's GPIO. The Pi's GPIO lines are not buffered so shorting them or applying voltages outside the specified limits could result in damage to the Raspberry Pi. For more information on the Pi's GPIO please follow this LINK to the official Raspberry Pi site.

Pi-LD is intended as an electronics / software development project, in connecting your Pi-LCD pledge reward to a Raspberry Pi computer you are accepting that the supplier / designer of the Pi-LCD cannot be held liable for any consequential damage or losses how so ever caused.

The Pi-LCD has been tested extensively with the Pi A+, B+ and Pi B Model 2. It is not suitable for use with the original A or B versions, these are easily identified as they have the smaller 26 way GPIO connector. There is no reason why it will not work with a PiZero but I have not tried it.

Contents

2 Quick start

Ok I know you just want to see your Pi-LCD working but, before we start please to take a moment to read through the do's & don'ts on page 1.

This section assumes you have a built Pi-LCD board and simply takes you through how to get your Pi-LCD working in the basic demonstration software.

2.1 Connecting your Pi-LCD

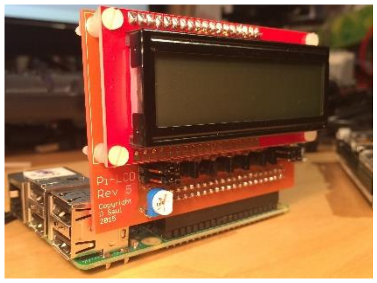


Figure 1 board orientation

- Ensure your Raspberry Pi is connected to the internet
- Download the *Helloworld* file from Pi-LCD GitHub DMS_PiLCD to your Pi (for more on downloading software see section 6.1)
- Shutdown your Pi and disconnect the power supply
- Ensure there is nothing else connected to the GPIO connector
- Carefully insert the Pi-LCD board into the GPIO connector so the board front is facing away from the Pi – see figure 1, making sure that board and Pi connectors are correctly aligned.
- Reconnect power to the Pi
- When it has booted up navigate to the directory you downloaded the *helloworld* file to
- Start the *helloworld* program
- The demo s/w should start immediately, displaying the time and a welcome message
- **If you cannot see anything you may need adjust the contract potentiometer**

3 Build Instructions

If you opted for the bare PCB pledge award the full parts list can be found in the appendix [9.3].

Preparation

Please read through all the instructions before you start. The instructions assume a basic level of soldering competence, if you are in any doubt there is lots of material on web to help you – one example on the Adafruit site is - adafruit-guide-excellent-soldering

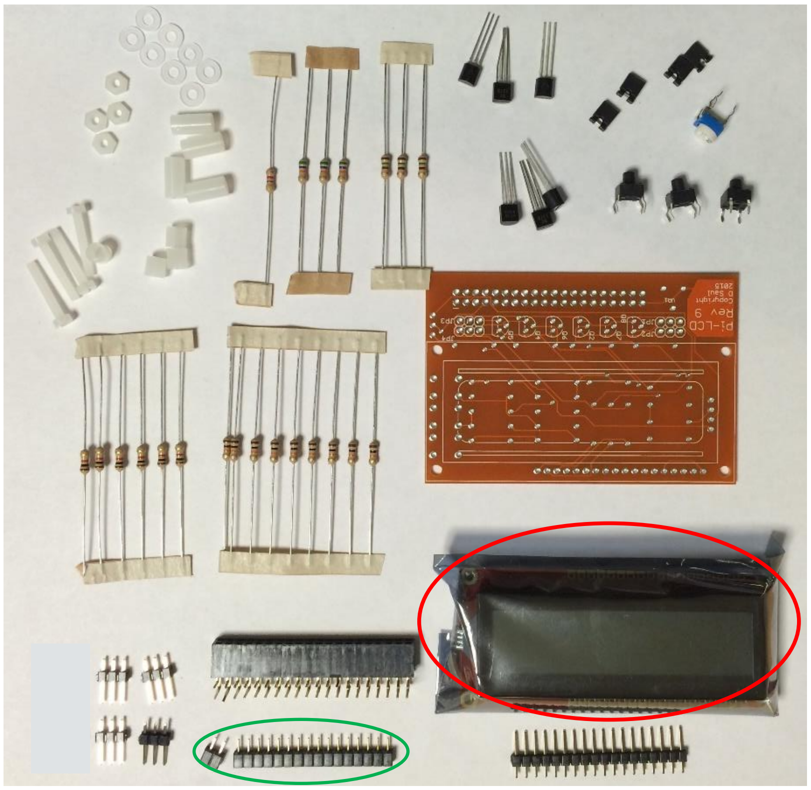
- There are no large areas of copper, so a 25W or less soldering iron will be fine
- When handling the LCD ensure you take static precautions
- Make sure you fit the parts to the correct side of the PCB, the part is always fitted to the same side as it's silk screen decal
- Be careful to fit the resistors in correct place
- Make sure you get the transistors in the correct places - it does matter!
- If you need to force anything you are trying to put it in the wrong place!

1 **Part Check**

Lay all the parts out to confirm you have everything.

Notes :

- The red circled item is only supplied with kits that include an RGB LCD display
- The green circled item is the low profile LCD socket where requested in your survey response this will be swapped for a 'normal' height socket



3

Resistors

Fit and solder resistors to the rear of the PCB;

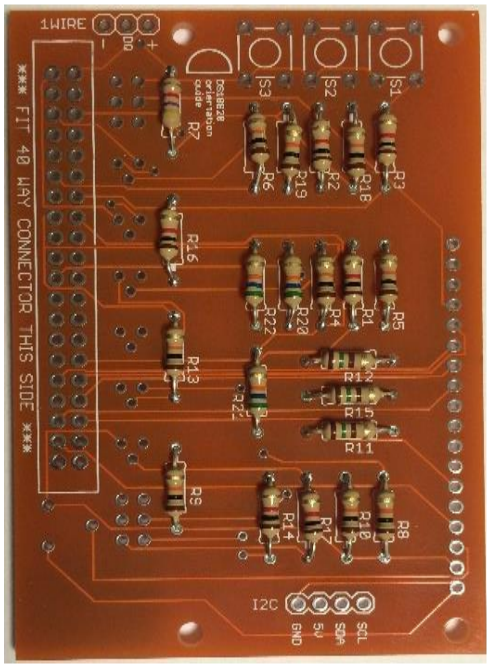
1k - R1,R2,R3,R14,R16,R17

10k - R4,R5,R6,R8,R9,R10,R13,R18,R19

4k7 - R7

150R - R11,R12,R15

56k - R20,R21,R22



2

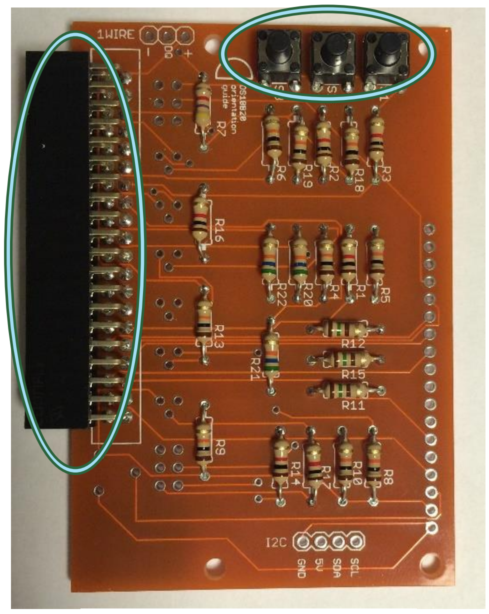
40 way connector, pushes

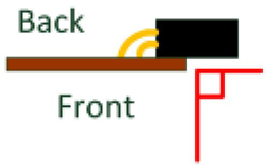
Solder the 40 way Pi GPIO connector to the **<u>REAR side of the PCB</u>**. To ensure the Pi-LCD board stands perpendicular to the Pi when complete it is important to make certain the connector is fitted square to the PCB [see diagram to right]. To do this firstly make sure the PCB is supported then solder a single corner pin first, check the alignment carefully re-soldering if needed. Then solder the opposite corner pin and again adjust the alignment by re-melting the solder if needed. Once you are happy that it is square solder the remaining pins.

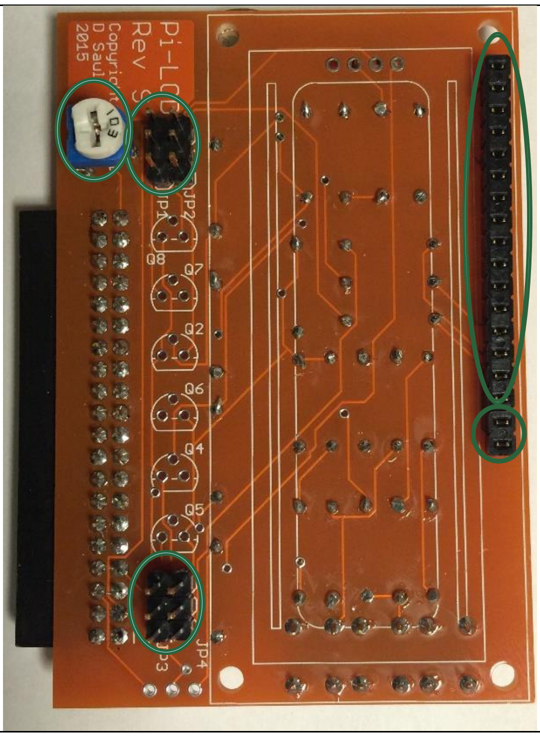
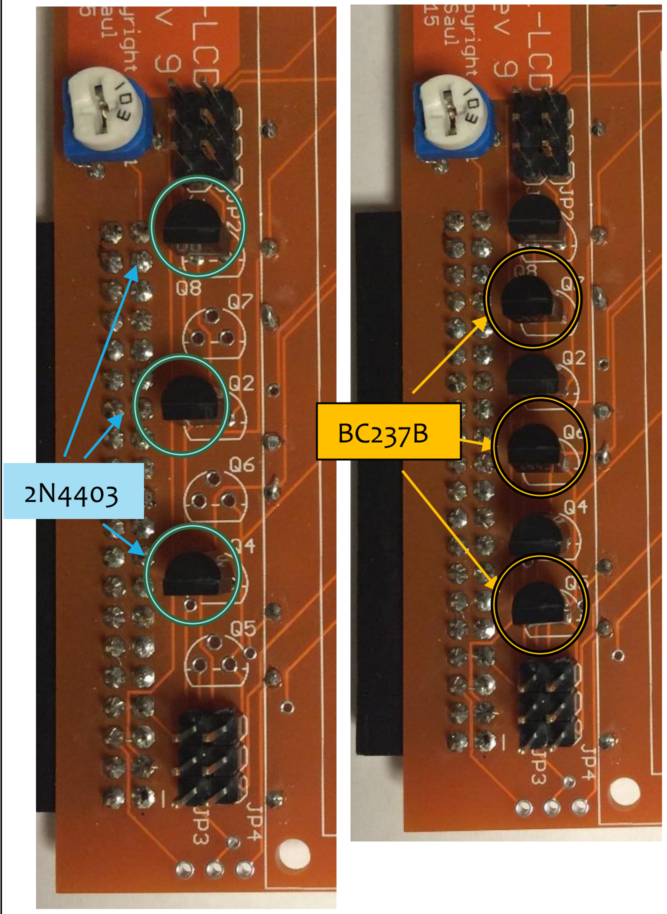
Also fit the 3 off pushes S1,2,3 at this stage

This completes the components which are fitted the rear of the PCB

[Note headers for the I2C and 1wire interfaces are not supplied with any of the pledge options]





4	**LCD socket, Pot and headers** Fit the LCD socket first, if you have to the low profile option this will be supplied as a 16 way and 2 way connector. Next fit the 10K pot and finally the 4 off 3 way headers	
5	**Transistors** Finally fit the transistors, these have very small writing on them so to help identify the 2N4403 transistors I have put a purple mark on the top of the case. It is important to get the transistors in the correct position as one is a PNP the other NPN 2N4403 [look for purple mark] - Q2,Q4,Q8 BC237B - Q5,Q6,Q7	

6

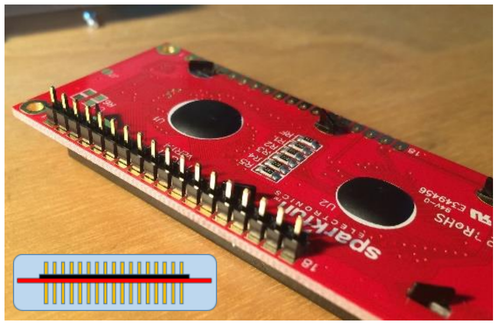
Display header pins

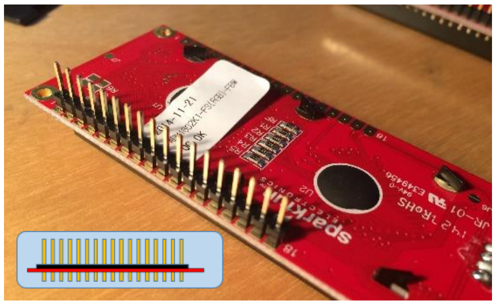
This assumes you are using the RGB display supplied as part of the full kit, if you are providing your own display you may need to adapt accordingly.

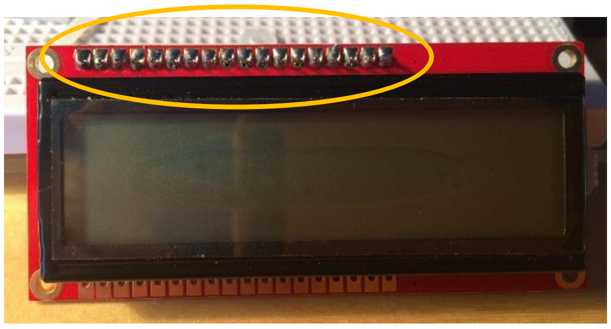
Most people have opted for the low profile connector, in this case you need to solder the header as shown in the top diagram [short pins coming away from the LCD sub-board]. For 'normal' height headers refer to the middle picture [long pins coming away from the LCD board].

It is worthwhile doing a trial build before you start soldering if you are in a doubt about this step

IN BOTH CASES MAKE SURE YOU SOLDER THE HEADER TO THE CORRECT PLACE ON THE LCD SUB-BOARD AS SHOWN IN THE BOTTOM PICTURE







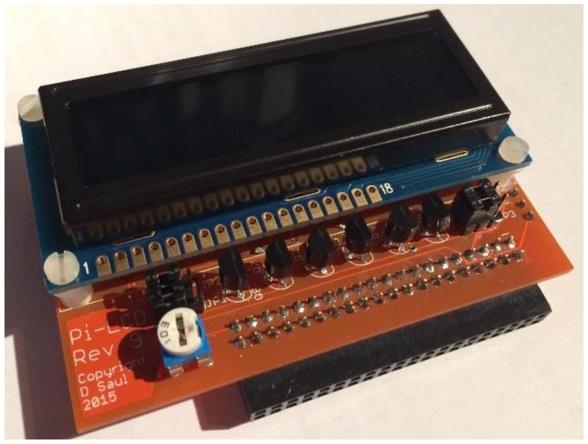
Mechanical fixing and jumpers

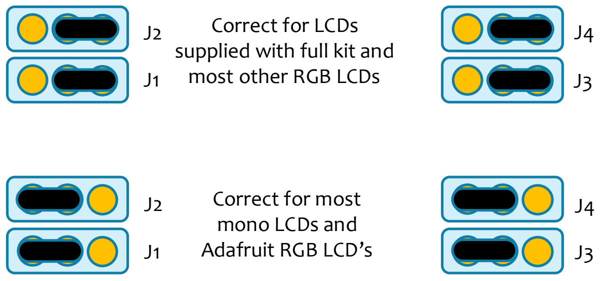
Two different sizes of spacers together with washers are supplied with the kits, this allows for a range of stand-off heights.

These worked for me

- low profile, 5mm + 2 washers
- normal profile, 10mm + 2 washers

Be careful not to over tighten the fixings as you could damage the LCD board if your spacer is lower than the connector stand-off



8	**Jumper-Links** There are 4 headers on the Pi-LCD board these allow common cathode and anode backlight LEDs to be used. Fit the 4 shorting links as per the instructions to the right.	  J2 J1 Correct for LCDs supplied with full kit and most other RGB LCDs J4 J3 J2 J1 Correct for most mono LCDs and Adafruit RGB LCD's J4 J3 Viewed from the front
9	**Visual check** Your Pi-LCD is now completed and ready to test, but before you connect it to your Raspberry Pi visually check the completed board against the part list shown section 9.3 to check there no parts in the wrong place, dry joints or no solder shorts.	

4 Connecting the Pi-LCD to your Raspberry Pi

No modifications to your Pi board are needed to connect the Pi-LCD, but obviously if it is in a box you will need to be able to access the full GPIO connector.

To connect the Pi-LCD;

- *Shutdown your Pi and disconnect the power supply*
- *Ensure there is nothing else connected to the GPIO connector*
- *Making sure the bottom the Pi is well supported, carefully insert the Pi-LCD board into the Pi GPIO connector so the board front is facing away from the Pi – see figure 1*
- *Make sure that board / Pi connectors are correctly aligned and that the Pi-LCD is fully seated into the Pi's GPIO connector*
- *Reconnect power to the Pi*

5 Raspberry Pi Setup

5.1 Assumptions

These instructions assume you are using the *RASPBIAN* operating system as supplied in the *NOOBS install version 1.4 or later*. The latest image they have been tested against is version 1.5.0 dated Nov 2015.

There have been a number of incremental improvements / changes to the *RASPBIAN* operating system since its initial release. That said in its basic form the Pi-LCD should work with any version dating from the release of the B+ in July 14.

To keep everything simple the example software uses the default *RPi.GPIO* library.

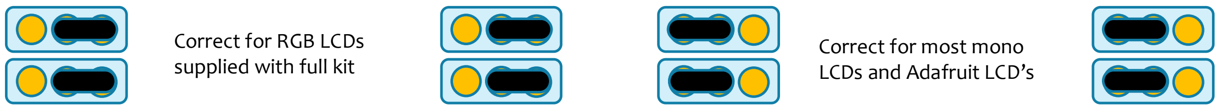
The instructions are written on the basis that you have no other specialist GPIO functionality implemented.

As with all software nothing stays the same for very long. The plan is to provide updates to these instructions via the Pi-LCD GitHub DMS_PiLCD to cover any significant changes to *RASPBIAN* that could affect operation / setup of the Pi-LCD until at least late 2016. It is however always a good idea to keep an eye on the Raspberry Pi forum for discussion on changes / developments to the operating system.

5.2 Basic Operation

5.2.1 Board links

There are 4 link setting on the Pi-LCD pcb these are used to select between common cathode and common anode displays. They are either set all to the left or all to right as shown in the pictures below


Correct for RGB LCDs supplied with full kit
Correct for most mono LCDs and Adafruit LCD's

Viewed from the front

If you get the links the wrong way round the LCD backlight will not work correctly but it will not damage the LCD or board.

5.2.2 Contrast pot

You will need to adjust the contrast pot depending on your viewing angle – note if this is set incorrectly you may not be able to see anything on the display.

5.2.3 Raspberry Pi

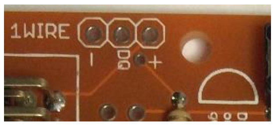
There is no additional system setup needed on your Pi for basic operation you can go straight on to section 6.1 - sample software.

5.3 Configuring the 1 Wire interface

The 1-wire interface is used to read the Ds18B20 temperature sensor sent as an additional reward to Kickstarter backers connects Pi-LCD on GPIO 04 line, via the '1wire' connector.

Assumptions

- Clean NOOBS install [1.4 or above] and The Raspberry Pi Connected to internet by cable or wireless
- You have connected DS18B20 to the Pi-LCD board as indicated on the PCB
- You have not already enabled the 1-wire interface



[WARNING:- fitting the DS18B20 the wrong way around will destroy it]

From the console or a *terminal* window in the graph user interface [GUI] enter the following commands.

	Action	***Command(s) to enter***
1	Ensure you are running latest s/w	*sudo apt-get update* *sudo apt-get upgrade*
2	Download the '*helloworld*' program	See 6.1
3	Run *helloworld app*, to confirm hardware is connected ok	*sudo python seq_test_2_r1.py*
4	Stop the application	hit '*control + c*'
5	Edit config.txt to enable DT support for 1-wire interface	*sudo nano /boot/config.txt*
6	Add '*overlay=w1-gpio*' command	At the end of the config.txt file add '*dtoverlay=w1-gpio*' Save and exit the editor
7	Setup the 1-wire interface to start at power up, and restart	*sudo modprobe w1-gpio* *sudo modprobe w1-therm* *sudo reboot*
8	Check it is all working, after entering the 'ls' command you should see the following response `28-0000058d6e88  w1_bus_master1` **Note** the number following the 28- will be different. This is the DS18B20's id and is unique to an individual sensor	*cd /sys/bus/w1/devices* *ls*
9	Change to the sensor sub-directory	*cd 28-xxxxxxxxxxx* [replace xxxx with your specific sensors id]

10	Check the readback
	You should see something similar to this,
	`b0 01 4b 46 7f ff 10 10 3a : crc=3a YES`
	`b0 01 4b 46 7f ff 10 10 3a t=27000`
	The 't' value is the measured temperature in thousands of the degree centigrade. The 'YES' indicates a valid reading
		cat w1_slave

11	The 1-wire interface is now enabled and running on GPIO 04, it will auto start at power up	

Table 1, 1-wire setup

5.4 I2C interface

The Pi-LCD also includes PCB pads to enable easy connection of I2C device, please refer to installation instructions for your specific device.

WARNING The I2C interface provides a connection to the +5V rail. If used to power your I2C device it is important to check that any pull-up resistors will NOT pass 5V to the SCL and SDA lines as this could damage you Raspberry Pi. Note – there is a 3.3V connection available on the 1wire interface.

6 Supplied Sample Software

6.1 Downloading Pi-LCD sample software

The sample software can be downloaded from the Pi-LCD GitHub DMS_PiLCD - To save the required files to your computer using the Pi's epiphany browser right-click on the filename and select '*save link as*'.

Software naming convention

Example software is available for Python 2 and Python 3 the file naming convention is

*title*_ *python version*_**r***revno***.py**

where :-

- ***title*** = software name, see 6.3 on
- ***python version*** = 2 or 3 [where there are separate applications]
- ***revno*** = integer number starting a '1' to differentiate updates to sample s/w code

For example if you are looking for the basic 'hello world' python code described in 6.3 to run in Python version 2 the file name will be:- *helloworld_2_r1.py*

For applications that work directly with Python 2 or 3 from a single app the naming is:- *RSSdemo_r1.py*

[Remember that file names are case sensitive in Linux]

6.2 Sample applications, overview

The sample applications have been purposely written with the aim that they are relatively easy for people with limited knowledge of Python to understand. Duplicate versions coded for Python 2.xx and 3.xx are available for down load. If you are using the latest 'JESSIE' version of RASBIAN [dated Sept 15, Kernel 4.1 or later] the applications can be run without 'Administrator' rights, for example

python helloworld_2_r1.py

However if you are any of the RASPBIAN WHEEZY versions [Kernel 3.18 or earlier], you will need to run all the demo software with 'Administrator' rights, for example

sudo python helloworld_2_r1.py

You can check you version by holding down the shift key when you re-boot your Pi, this will take you to the set-up screen and you can easily see if you are running WHEEZY or JESSIE by looking at the RASPIAN row. Hit 'Esc' to then reboot as normal.

The demo programs have all be tested running from the console and in a terminal window, but should also work fine through 'IDLE'

The sample applications were developed and tested on Python 2 version 2.7.3 and Python 3 version 3.2.3 .

The following demo applications / software will be available from the GitHub DMS_PiLCD initially.

- helloworld, basic test demonstration - standalone app
- scrolldemo, demo using threading to show two display update processes working together
- RGBdemo, demo to show the range of colours available with RGB backlit displays
- bigcharclock, demo to using custom characters to display and 'oversize' clock display
- RSSdemo, basic RSS feed demo application
- DMS_PiLCD Class:, Python class developed simplify applications written for the Pi-LCD board

  Check the GitHub for other applications

With the exception of "helloworld" the demo software all makes us of the DMS_Pi_LCD class, this requires DMS_Pi_LCD.py to be in the same subdirectory as the demo software being run.

Finally please do keep in mind that these are demonstration applications, they do have some rough edges and are certainly not developed to the point where errors will always be cleanly trapped.

6.3 Sample application detail

6.3.1 Helloworld

helloworld_2_r1.py - python 2.xx version
enhancedclock_3_r1.py - python 3.xx version

helloworld is a basic application to test the core features of the Pi-LCD board. Importantly *helloworld* unlike the other demo applications is standalone – it does use the *DMS_PiLCD* class subroutines.

When started, the screen will display a fixed test message and you will be prompted to enter your own text message [max 16 characters] . Once entered this is displayed on the lower line of the display with a clock on the top line. Back light colours can be selected using the 3 pushes on the rear of the board – note these are just on / off in this application.

6.3.2 scrolldemo

File name;

scrolldemo_2_r1.py - python 2.xx version
scrolldemo_3_r1.py - python 3.xx version

scrolldemo demonstrates the use of Python threading to do a couple of things at once in this case it simply prompts for you to enter a message, which is then displayed. The mildly [well in my view] clever bit is that the scroll thread continues while the main program code waits for you to enter new text, picking this up and updating the display message. As an added bonus each time you enter a new message the backlight is changes randomly.

6.3.3 RGBdemo

big_ck_2_r1.py - python 2.xx version
big_ck_3_r1.py - python 3.xx version

RGBdemo allows you to individually set each of the 3 back light LED colour intensity. The top push selects the display colour to change, middle push increases intensity the bottom push reduces it. The intensity value between 0 and 100 [FP] for each colour is displayed on the second line of the display.

6.3.4 bigcharclock

File name;
big_ck_r1.py - works with either python 2.xx or 3.xx

This application demonstrates using how you can use the 8 user defined characters available on HD44780 compatible displays to display an 'oversized' clock. Additionally if you have a DS18B20 fitted and push the top momentary switch it will display temperature for a second

[If a DS18B20 is not fitted it will display an error warning on the LCD for second and then revert to the clock display]

6.3.5 RSSdemo

File name;

RSSdemo_r1.py - works with either python 2.xx or 3.xx

This application downloads and displays a selection of BBC RSS news feeds on line 2, with time and date on line one.

To work RSSdemo you need to have installed 'feedparser' with the following instruction from the command line

For Python 2 - *sudo pip install feedparser*

For Python 3 – *sudo pip3 install feedparser*

Obviously your Raspberry Pi will need to be connected to the internet for this application to work as intended.

There are 2 routines you may find useful in other applications in RSSdemo

`getrss(subject)`	`RSS function - get rss text with some limited handling for an IO exceptions`
`mess_build()`	`RSS function - build message from individual rss feeds, used to build a single string containing a number of RSS feeds`

6.4 DMS_PiLCD Class description

This section provides an overview of the hardware definitions and sub-routines in the DMS_PiLCD class. For detailed information please see the commentary embedded in the code listing.

6.4.1 Hardware definitions

The following list details the hardware definitions used by the Pi-LCD sub-routines

GPIO to LCD mapping [BCM GPIO numbers]

```
LCD_RS = 20   # pin 38
LCD_E  = 19   # pin 35
LCD_D4 = 16   # pin 36
LCD_D5 = 13   # pin 33
LCD_D6 = 12   # pin 32
LCD_D7 = 6    # pin 31

LCD_LEDR = 18 # pin 12
LCD_LEDG = 22 # pin 15
LCD_LEDB = 23 # pin 16
```

GPIO to user switch mapping

```
LCD_SW1 = 27 # pin 13
LCD_SW2 = 25 # pin 22
LCD_SW3 = 5  # pin 29
```

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
LCD_LINE_3 = 0x94 # LCD RAM address for the 2nd line - 4x20 line disp only
LCD_LINE_4 = 0xd4 # LCD RAM address for the 2nd line - 4x20 line disp only
```

Timing constants

```
E_PULSE = 0.0005
E_DELAY = 0.0005
```

GPIO setup

```
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
GPIO.setup(LCD_LEDR, GPIO.OUT)    # Backlight
GPIO.setup(LCD_LEDG, GPIO.OUT)    # Backlight
GPIO.setup(LCD_LEDB, GPIO.OUT)    # Backlight
GPIO.setup(LCD_SW1, GPIO.IN)      # switch 1
GPIO.setup(LCD_SW2, GPIO.IN)      # switch 2
GPIO.setup(LCD_SW3, GPIO.IN)      # switch 3
```

PWM channels definitions

```
pr = GPIO.PWM(LCD_LEDR, 60)       # channel= LEDR  frequency=50Hz
pg = GPIO.PWM(LCD_LEDG, 60)       # channel= LEDG  frequency=50Hz
pb = GPIO.PWM(LCD_LEDB, 60)       # channel= LEDB  frequency=50Hz
```

6.4.2 Software sub-Routines

This section contains a list of the sub-routines you will find in the DMS_PiLCD Class, together with a brief description for more refer to the code listing which is well commented. Check the GitHub for updates to this list [note the file name is *DMS_PiLCD1.py*].

<u>Low level standard functions for Pi-LCD communications</u>

These are used locally by the PiLCD class for LCD serial comms. There should be no reason to call them directly from your programs,

`lcd_byte(bits,mode)`	Send byte to data pins - using 4 bit nibble mode
`lcd_line_convert(line)`	convert line 1-4 to LCD hex RAM address for relevant line start
`lcd_toggle_enable()`	Toggle enable line on LCD used to latch data into LCD

<u>Main user routines to send characters to the LCD</u>

`lcd_init()`	Initialise display, this must be called before any other commands are be sent to the LCD
`lcd_defchar(char_no,char)`	define 1 of the 8 available user defined characters
`lcd_setcursor(col,row)`	set cursor to column / row
`lcd_char(char)`	send character [in range 0-255] to display at current cursor position
`lcd_string(message,line,pos)`	Send string to display on 'line' at 'pos'
`lcd_scroll(speed,line)`	Special function see 6.4.3 for description
`lcd_cls(back_lite)`	Clears the display and optionally turn off the backlight

<u>PWM backlight control routines</u>

These use the new'ish PWM function in the RPi.GPIO class

`start_PWM()`	Start 3 PWM Ports, this Routine has to be run before any of the other sub-routines using PWM if you want to control colour levels with PWM
`led_set(ledr,ledg,ledb)`	Change Colour intensity levels
`led_set_colour(bl_col)`	set display to one a 10 predefined colours

<u>**Non** PWM backlight control routines</u>

These provide 7 basic colours, without the use of PWM in the RPi.GPIO class. Because there is no way to balance LED intensities the yellow and white do not look great.

`nonPWM_led_set_colour(bl_col)`	set display to one a 10 predefined colours (white,red,green,blue,yellow,magenta,cyan)
`nonPWM_led_set_num(col)`	set display to one a 10 predefined colours, with integer in the range 0 to 6 (0=white,1=red,2=green,3=blue,4=yellow,5=magenta,6=cyan)

Oversize Character routines

These simplify setting up and displaying oversized numeric character displays

`overchar_setup()`	`Set-up user Characters, this is called initially to set-up user defined characters for oversize numeric display`
`lcd_disp_ec(col,char)`	`Display oversized character starting at specified column pos note each Char take 3 columns`

Extra stuff - not LCD specific

These are extra routine which are used by some of the demo applications

`sens_add()`	`DS18B20 function - Identify folder address for DS18B20`
`read_temp_file(device_file)`	`DS18B20 function - Get raw temp data`
`get_temp(id):`	`DS18B20 function - Return scaled temp, this is the main user function needed to readback temperatures`

6.4.3 Scroll Routine

`lcd_scroll` provides a scrolling text function which permits concurrent scrolling with other display information updates. It is invoked using the Python 'threading' function rather called as a normal function. As written the thread will never exit. There are 2 variables associated with `lcd_scroll;`

`mess,` this is a string variable which contains the text you want to display as a scrolling message on the LCD. Any changes to this will be picked up by `lcd_scroll` and the displayed message updated

`bflag,` this is a flag and is used to temporally freeze the scrolling. This is used whenever you want to call a routine which sends information the LCD. It is needed to avoid clashes between the main and threaded program elements sending data or commands to the LCD. To keep the scrolling smooth it is important to keep the time bflag is set to a minimum

The best way to understand how `lcd_scroll and the DMS_PiLCD` class routines work is to look at the *scrolldemo* application.

6.5 Your own Python coding

If you want to develop your own code for the Pi-LCD you can either start with one of the sample application or if you want to start from scratch you just need to include the following lines of code to access the routines and Pi-LCD hardware definitions and software routines described in 6.4.

From DMS_PiLCD1 import DMS_PiLCD
PiLCD = DMS_PiLC*D()*

7 Using your Pi-LCD board

The best way to understand what you can and cannot do with your PiMuxClock is to try out the sample applications and go on from there, these are described in section 6.

For details on addressing the LCD display please check the HD44780 datasheet* – this can be downloaded from the GitHub. [Note : -This is the datasheet for the original Hitachi silicon, most clones follow this closely but you may find some differences between character maps and support for larger displays].

8 Problems

The Pi-LCD design is quite straight forward so hopefully you should not have any problems, the following list is based experience gained during developing testing the board and s/w.

Pi will not start with Pi-LCD connected

- *Check you have connected the Pi-LCD correctly and fully seated into the Pi's GPIO connector*

You get a runtime error when you try to run any of the applications

- *Make sure you run the application with root privileges - i.e. 'sudo python hello…..' , if you are using WHEZZY*
- *Make sure you have DMS_PiLCD1.py in the same sub-directory as the python application you are running*

Pi-LCD does work at all

- *Built from kit, check against section 3 to be sure that you have soldered everything in the correct place*
- *Check you have adjusted the contrast pot correctly*

The display backlight not working

- *Check* you have configured the jumpers correctly

RSS application does not work

- *Make sure you have feedparser installed correctly* [*see 6.3.5*]

DS18B20 seems to display temperatures higher than ambient

- *If you have soldered the DS18B20 directly to the PCB you will find it picks up conducted heat from the Raspberry Pi causing it to apparently 'over read' the room temperature. This happens because the actual temperature element is bonded to the middle leg of the chip. Mounting it on short wires to bring it away from the board should ensure it displays the room temperature accurately.*

9 Appendix, Technical Information

9.1 GPIO assignments

See section 6.4.1

9.2 I2C and additional GPIO connections

I2C and 1-wire connections are clearly labelled PCB silk screen

9.3 Parts List

Parts are available from a number of suppliers, although you may need to shop around for the 40 way right-angle connector. Between them all the parts can be purchased from Bitsbox and Hobbytronics in the UK, with the exception of the low profile LCD sockets. These can be sourced from Toby Electronics.

Ref	**Part Description**	**Quantity**
PCB	Rev 9 PCB	1
LCD Header	1X18 header	1
LCD socket *	1X18 header socket - NORMAL PROFILE	1
LCD socket - made up of 2 & 16 way parts *	1X18 header socket - LOW PROFILE	1
40 way RPi Connector	20+20 right angle connector	1
JP1,JP2,JP3,JP4	1X3 header	4
Q2,Q4,Q8	2N4403 [look for purple mark on top]	3
Q5,Q6,Q7	BC237B	3
R1,R2,R3,R14,R16,R17	1k	6
R4,R5,R6,R8,R9,R10,R13,R18,R19	10k	9
R7	4k7	1
R11,R12,R15	150R	3
R20,R21,R22	56k	3
VR1	10k Trimmer pot	1
S1,S2,S3	MOMENTARY-SPST NO	3
5mm spacer	nylon spacer 5mm	4
10mm spacer	nylon spacer 10mm	4
fixing	nylon nut and bolt 20mm / 16m mm2.5	4
washer	m2.5 nylon washer	8
shorting link	Jumper shorting 2 way	4
LCD	16x2 LCD colour	1
Temp Sensor	DS18B20	1
* depends on survey response		

Table 2, part list

Points to watch if you are trying to purchase replacement parts;

- *The display meets the mechanical interface requirement [details on Kickstarter and on the Pi-LCD GitHub DMS_PiLCD]*
- *Make certain you are buying a right angle [90^O] 40 pin connector*
- *Make sure the momentary switches as normally open types*