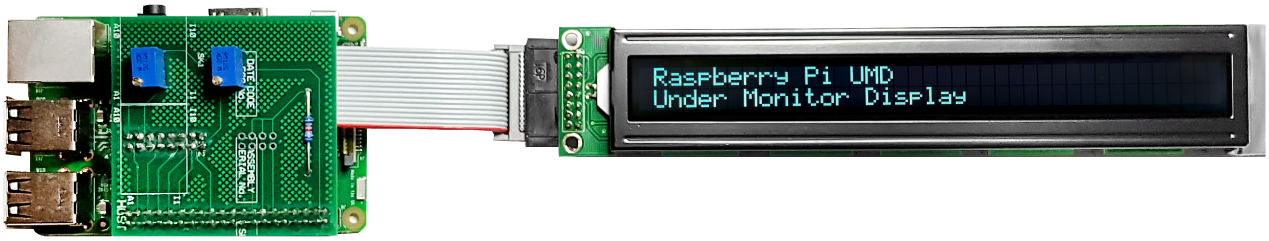
The Python Code



The Code Running the LCD

By now you should have your Raspberry Pi and LCD running. Here we will discuss the Python code that runs the LCD display and how it works (at a top level).

The Code

On each Raspberry Pi there is a file called UMDisplay0x.py. This is the file that sends data to the LCD, it is the same on each Pi except for a setting that tells the Pi which 2 of the 8 text lines to use.

(This is from the Master Pi)

The File Header

This is to identify to the Linux system what this file type is, and to allow the author to add some information (anything following a hash # is just a comment)

```
#!/usr/bin/python
#--------------------------------------
#  16x2 LCD Test Script
#
# Author : Alan Walker
# Date   : 16/05/2016
#--------------------------------------
```

A reminder of the pin outs

The following section is just comments, but it is a nice reminder of which GPIO pins on the Raspberry Pi that this Python code is going to try to address.

```
# The wiring for the LCD is as follows:
# 1 : GND
# 2 : 5V
# 3 : Contrast (0-5V)*          -   Sets the LCD Contrast
# 4 : RS (Register Select)      -   GPIO pin 26 - Physical pin 37
# 5 : R/W (Read Write)          -   GROUND THIS PIN
# 6 : Enable or Strobe          -   GPIO pin 19 - Physical pin 35
# 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                -   GPIO pin 13 - Physical pin 33
# 12: Data Bit 5                -   GPIO pin 06 - Physical pin 31
# 13: Data Bit 6                -   GPIO pin 05 - Physical pin 29
# 14: Data Bit 7                -   GPIO pin 11 - Physical pin 23
```

```
# 15: LCD Backlight             -    +5V**
# 16: LCD Backlight             -    GND
```

Import Python Libraries

Python uses standard libraries to provide standard functions, so you don't have to code everyday tasks such as printing etc. Here we are importing some standard libraries for our project.

```
#import
import RPi.GPIO as GPIO
import time
import socket
import fcntl
import struct
```

Define some Constants

Rather than referring to GPIO pin numbers in the code, we are declaring some variables that we can use to reference the GPIO pin numbers to actual functions. (So the LCD Reset that uses GPIO pin 26 can be called by using just LCD_RS for example).

The advantage of using this method is, if a pin number changes, we can change it here, and not have to hunt through the code for every occurrence of that pin number.

```
# Define GPIO to LCD mapping
LCD_RS = 26
LCD_E  = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11
```

Import Python Libraries

Below we are defining some more constants, these are to do with the LCD lines and chars, as well as the timing.

```
# Define some device constants
LCD_WIDTH = 40    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

# Address for 40 char display
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
```

Code Start

This is where the main code starts, as this program runs in a loop, anything above the line def main(): is only run once at program start. Anything below is run once per loop.

GPIO.setmode(GPIO.BCM) tells the Pi we are using the GPIO board numbers, not the Pin numbers.

```
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
```

Program Start

This is the main body of the code that runs the LCD, to save splitting this up I will comment in this section.

```
#########################################
#             PROGRAM  START             #
#########################################


  # Initialise display
  lcd_init()

  # If using DHCP this is required, or code runs before network is ready.
  lcd_string(" Waiting 5 Seconds for Network Start",LCD_LINE_1) # wait 5 seconds for DHCP process to
finish
  time.sleep(5)
```

Read File

Here we are opening the file '/home/pi/Python/my_data.txt for edit (so we can read and write to it). This file contains the 8 lines of text for our LCDs.

```
  while True:
    ## Open the file with read only permit
    f = open('/home/pi/Python/my_data.txt', "r")

    ## use readlines to read all the lines in the file
    ## The variable "lines" is a list containing all lines
    # readline 400 is read in 400 chars
    # replace removed the carriage return and new line
    # [:40] truncates to 40 chars

    line1 = f.readline(400).replace("\r\n","")[:40]
    line2 = f.readline(400).replace("\r\n","")[:40]
    line3 = f.readline(400).replace("\r\n","")[:40]
    line4 = f.readline(400).replace("\r\n","")[:40]
    line5 = f.readline(400).replace("\r\n","")[:40]
    line6 = f.readline(400).replace("\r\n","")[:40]
    line7 = f.readline(400).replace("\r\n","")[:40]
    line8 = f.readline(400).replace("\r\n","")[:40]

    ## close the file after reading the lines.
    f.close()
```

Set which line to write to the LCD

The code below sets what lines from the 8 lines are used (so Master Pi displays lines 1&2, the fourth Pi displays Lines 7&8.

line1,LCD_LINE_1

This above example tells the Pi to use Line 1 from our text file, and put is on line 1 of our LCD

```
    # write line 1 and 2 to the LCD (Line7 and Line8 because this is UMD4)
    lcd_string(" " + line1,LCD_LINE_1)
```

```
    lcd_string(" " + line2,LCD_LINE_2)

    time.sleep(1) # x second delay
```

LCD Write Code

The code below takes our text and writes it to the LCD, do not make any changes to this section.

```
########################################
#     CHANGE NO CODE BELOW THIS LINE     #
########################################


def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])


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()
```

Complete Code Set

Below is the complete file without any breaks, you can cut and paste this to create your own UMD LCD.

```
#!/usr/bin/python
#--------------------------------------
#  16x2 LCD Test Script
#
# Author : Alan Walker
# Date   : 16/05/2016
#--------------------------------------

# The wiring for the LCD is as follows:
# 1 : GND
# 2 : 5V
# 3 : Contrast (0-5V)*          -   Sets the LCD Contrast
# 4 : RS (Register Select)      -   GPIO pin 26 - Physical pin 37
# 5 : R/W (Read Write)          -   GROUND THIS PIN
# 6 : Enable or Strobe          -   GPIO pin 19 - Physical pin 35
# 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                -   GPIO pin 13 - Physical pin 33
# 12: Data Bit 5                -   GPIO pin 06 - Physical pin 31
# 13: Data Bit 6                -   GPIO pin 05 - Physical pin 29
# 14: Data Bit 7                -   GPIO pin 11 - Physical pin 23
# 15: LCD Backlight             -   +5V**
```

```
# 16: LCD Backlight             -   GND

#import
import RPi.GPIO as GPIO
import time
import socket
import fcntl
import struct

# Define GPIO to LCD mapping
LCD_RS = 26
LCD_E  = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11


# Define some device constants
LCD_WIDTH = 40    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

# Do I need to change line 2 address for 40 char display?
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

########################################
#             PROGRAM  START            #
########################################


  # Initialise display
  lcd_init()

  lcd_string(" Waiting 5 Seconds for Network Start",LCD_LINE_1) # wait 5 seconds for DHCP process to
finish
  time.sleep(5)


  while True:
    ## Open the file with read only permit
    f = open('/home/pi/Python/my_data.txt', "r")

    ## use readlines to read all the lines in the file
    ## The variable "lines" is a list containing all lines
    # readline 400 is read in 400 chars
    # replace removed the carriage return and new line
    # [:40] truncates to 40 chars

    line1 = f.readline(400).replace("\r\n","")[:40]
    line2 = f.readline(400).replace("\r\n","")[:40]
    line3 = f.readline(400).replace("\r\n","")[:40]
    line4 = f.readline(400).replace("\r\n","")[:40]
```

```
    line5 = f.readline(400).replace("\r\n","")[:40]
    line6 = f.readline(400).replace("\r\n","")[:40]
    line7 = f.readline(400).replace("\r\n","")[:40]
    line8 = f.readline(400).replace("\r\n","")[:40]

    ## close the file after reading the lines.
    f.close()

#    Display the current IP Address information
#    lcd_string("  lo" + " " + get_ip_address('lo'),LCD_LINE_1)
#    lcd_string("  UMD 04",LCD_LINE_1)
#    lcd_string(" eth0" + " " + get_ip_address('eth0'),LCD_LINE_2)

#    time.sleep(2) # x second delay

    # write line 1 and 2 to the LCD (Line7 and Line8 because this is UMD4)
    lcd_string(" " + line1,LCD_LINE_1)
    lcd_string(" " + line2,LCD_LINE_2)

    time.sleep(1) # x second delay


########################################
#     CHANGE NO CODE BELOW THIS LINE     #
########################################


def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])


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()
```