Interfacing 16×2 LCD with Raspberry Pi

In the previous project of the Raspberry Pi Series, I have shown you how to blink an LED using Raspberry Pi and Python Program. Moving forward in the series, in this project, I'll show you the interfacing 16×2 LCD with Raspberry Pi.

In this project, you can see all the steps for Interfacing a 16×2 LCD with Raspberry Pi like circuit diagram, components, working, Python Program and explanation of the code.

Even though the Raspberry Pi computer is capable of doing many tasks, it doesn't have a display for implementing it in simple projects. A 16×2 Alphanumeric Character LCD Display is a very important types of display for displaying some basic and vital information.

The combination of Raspberry Pi and 16×2 LCD Display can be used many projects and applications.

A Brief Note about 16×2 LCD

A 16×2 LCD is one of the most popular display modules among hobbyists, students and even electronics professionals. It supports 16 characters per row and has two such rows. Almost all the 16×2 LCD Display Modules that are available in the market are based on the Hitachi's HD44780 LCD Controller.

Typically, a 16×2 LCD Module consists of 16 Pins. The pin description of the 16×2 LCD Display Module is shown in the following table.

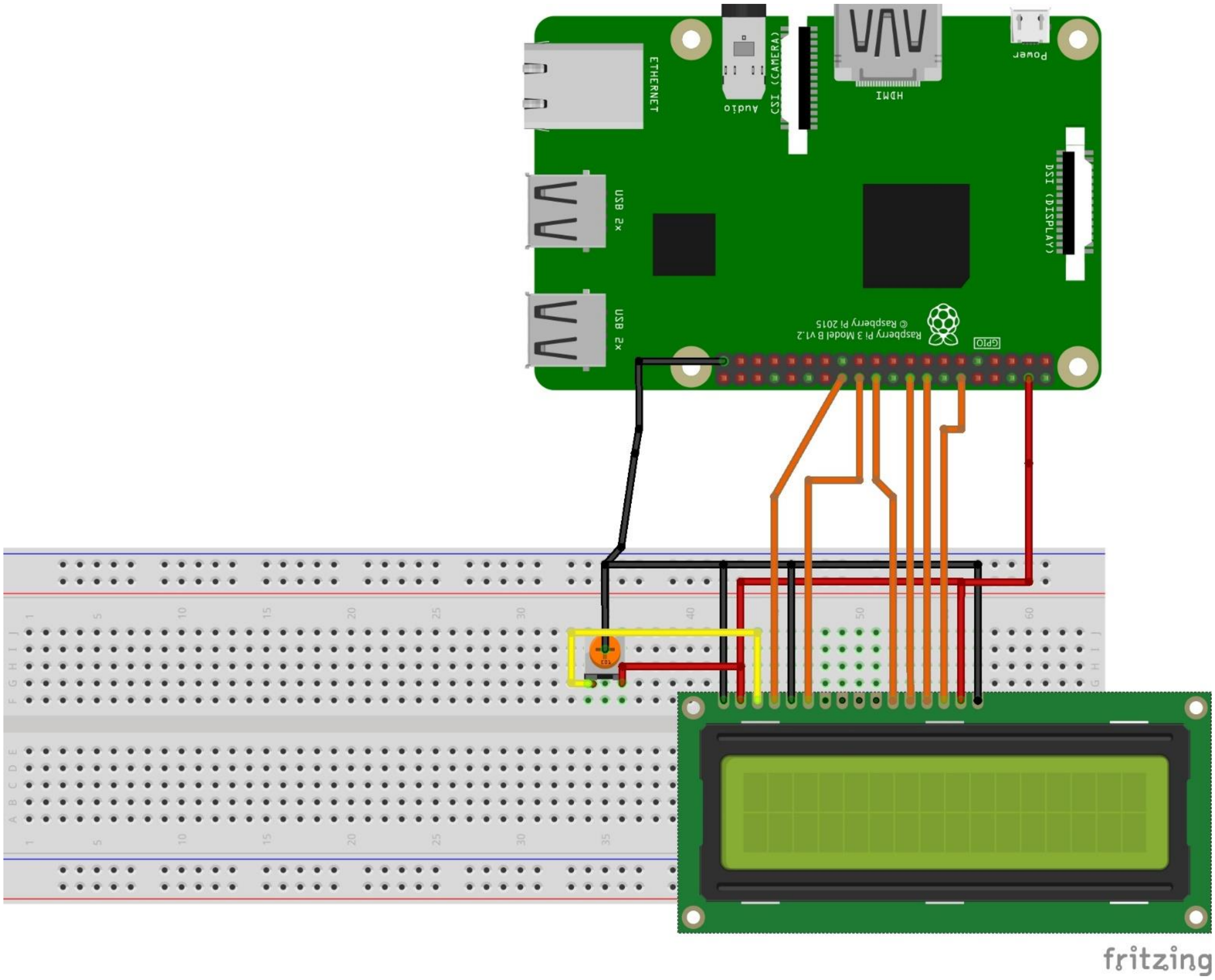
Pin No	Name	Function
1	VSS	GND
2	VDD	VCC(+5V)
3	VEE	Contrast adjust pin
4	RS	command register when 0; and data register when 1
5	R/W	0 to write; 1 to read
6	EN	Sends data to data pins when a high lo low pulse is given
7	DB0	Data pin
8	DB1	Data pin
9	DB2	Data pin
10	DB3	Data pin
11	DB4	Data pin
12	DB5	Data pin
13	DB6	Data pin
14	DB7	Data pin
15	LED+	LED Backlight (+5V)
16	LED-	LED Backlight (GND)

Circuit Diagram of 16×2 LCD Interfacing with Raspberry Pi

The pin description in the above table shows that a 16×2 LCD has 8 data pins. Using these data pins, we can configure the 16×2 LCD in either 8 – bit mode or 4 – bit mode. I'll show the circuit diagram for the 4 Bit mode.

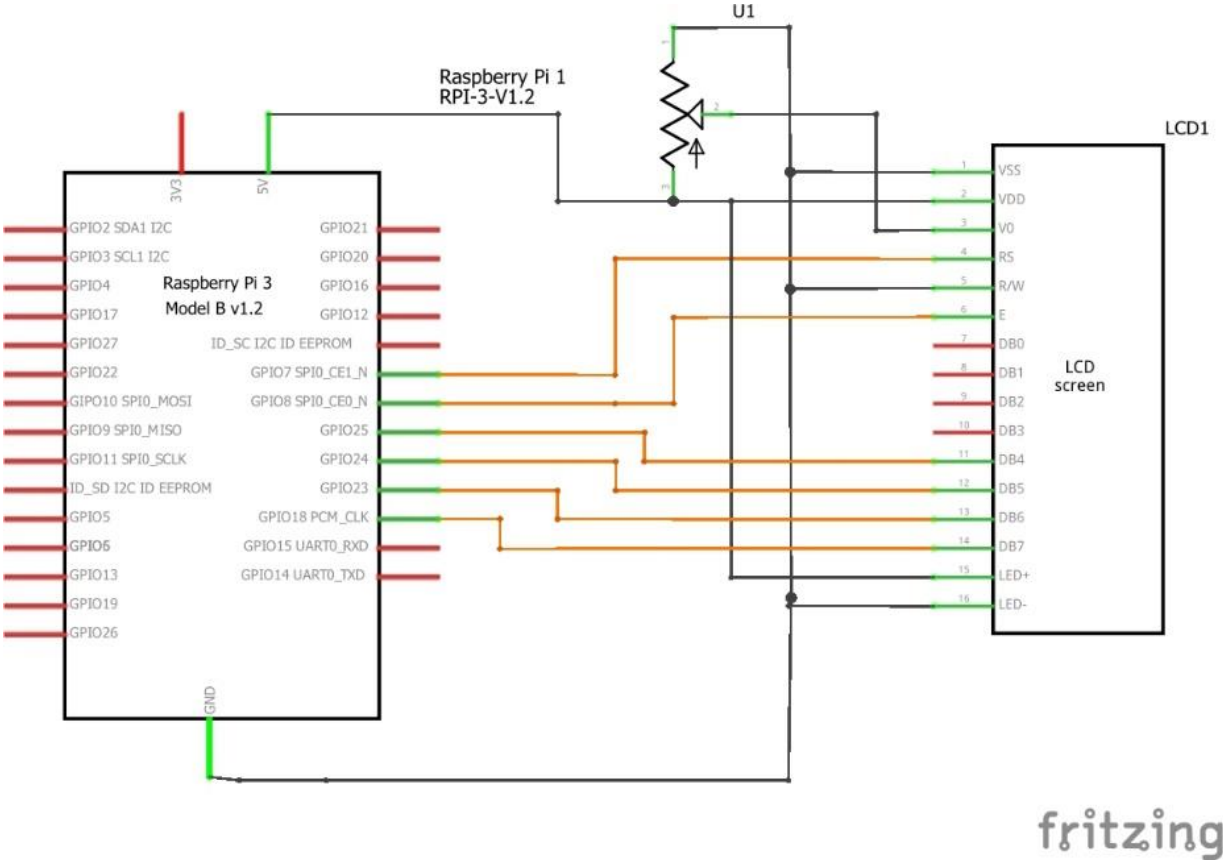
Fritzing Circuit

In 8 – bit mode, all the 8 data pins i.e. D0 to D7 are used for transferring data. This type of connection requires more pins on the Raspberry Pi. Hence, we have opted for 4 – bit mode of LCD. The circuit diagram (with Fritzing parts) is shown below.



Circuit Diagram

The following image shows the wiring diagram of the featured circuit of this project i.e. LCD in 4 – bit mode. In this mode, only 4 data pins i.e. D4 to D7 of the LCD are used.



Components Required

- Raspberry Pi 3 Model B (any Raspberry Pi)
- 16 x 2 LCD Module
- 10 KΩ Potentiometer
- Mini Breadboard
- Connecting wires (Jumper wires)
- 5V – 2A Power Supply
- Miscellaneous (Computer, Ethernet Cable, etc.)

Circuit Design

The design of the circuit for Interfacing 16×2 LCD with Raspberry Pi is very simple. First, connect pins 1 and 16 of the LCD to GND and pins 2 and 15 to 5V supply.

Then connect a 10KΩ Potentiometer to pin 3 of the LCD, which is the contrast adjust pin. The three control pins of the LCD i.e. RS (Pin 4), RW (Pin 5) and E (Pin 6) are connected to GPIO Pin 7 (Physical Pin 26), GND and GPIO Pin 8 (Physical Pin 24).

NOTE: The numbering of the Raspberry Pi Pins is expressed using BCM Numbering Scheme.

Now, the data pins of the LCD. Since we are configuring the LCD in 4 – bit mode, we need only 4 data pins (D4 to D7). D4 of LCD is connected to GPIO25 (Physical Pin 22), D5 to GPIO24 (Physical Pin 18), D6 to GPIO24 (Physical Pin 16) and D7 to GPIO18 (Physical Pin 12).

Python Program for Interfacing 16×2 LCD with Raspberry Pi

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
from time import sleep

# 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("L-TECH LABS ",LCD_LINE_1)
    lcd_string("    Presents    ",LCD_LINE_2)

    sleep(3) # 3 second delay

    # Send some text
    lcd_string("Rasbperry Pi",LCD_LINE_1)
    lcd_string("16x2 LCD Test",LCD_LINE_2)

    sleep(3) # 3 second delay

    # Send some text
    lcd_string("1234567890*@$#%&",LCD_LINE_1)
    lcd_string("abcdefghijklmnop",LCD_LINE_2)

    sleep(3)

def lcd_init():
  lcd_display(0x28,LCD_CMD) # Selecting 4 - bit mode with two rows
  lcd_display(0x0C,LCD_CMD) # Display On,Cursor Off, Blink Off
  lcd_display(0x01,LCD_CMD) # Clear display

  sleep(E_DELAY)

def lcd_display(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_display(line, LCD_CMD)

  for i in range(LCD_WIDTH):
    lcd_display(ord(message[i]),LCD_CHR)

if __name__ == '__main__':

  try:
    main()
  except KeyboardInterrupt:
    pass
  finally:
    lcd_display(0x01, LCD_CMD)
    GPIO.cleanup()
```

Code Explanation

First, I’ve imported the RPi.GPIO Python Package as GPIO (here after called as GPIO Package) and sleep from time package. Then, I have assigned the pin for LCD i.e. RS, E, D4, D5, D6 and D7. The numbering scheme I followed is GPIO or BCM Scheme.

NOTE: I have also commented the Physical pin numbers of the corresponding Pins.

Then I have used some functions of the GPIO package like GPIO.setwarnings (False), GPIO.setmode (GPIO.BCM) and GPIO.setup().

Finally, using some own functions like lcd_init, lcd_string, lcd_display, etc. I’ve transmitted the data to be printed from the Raspberry Pi to the 16×2 LCD Module.

Applications of Interfacing 16×2 LCD with Raspberry Pi

By interfacing 16×2 LCD with Raspberry Pi, we can have a simple display option for our raspberry Pi which can display some basic information like Date, Time, Status of a GPIO Pin, etc.

Many simple and complex application of Raspberry Pi like weather station, temperature control, robotic vehicles, etc. needs this small 16×2 LCD Display.

Limitations

The 16×2 LCD Module can only display simple alphanumeric characters.

Even though some special characters and custom characters can be displayed, information which is graphic intensive cannot be displayed.