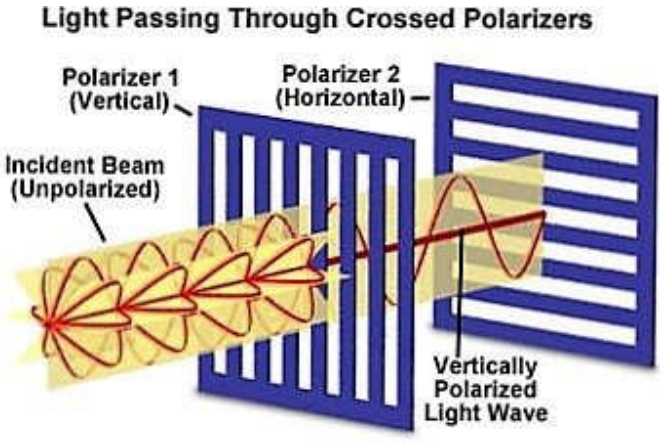
LAB MANUAL 4

Interfacing I2C Sensors

Practical Interfacing I2C Sensors

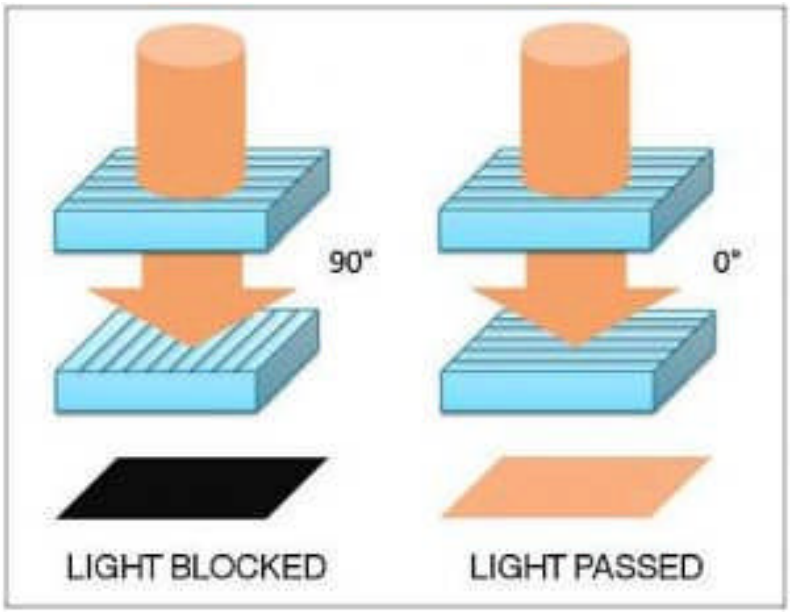
Liquid Crystal Display

LCD or Liquid Crystal Display is a flat electronic display which is very commonly used in digital watches, calculators, laptops, televisions etc. It make use of light modulating properties of liquid crystal and polarization of light for its operation. Low power consumption, less thickness and less weight of LCD enables its use in battery powered and portable applications.
The process of converting unpolarized light to polarized light is known as **Polarization**. Electric fields in polarized lights will vibrate only in a specific pattern (not randomly in all directions) or in one plane as given in the following diagram. Polarization can be done in different methods but LCDs use **Vertical and Horizontal Polaroid filters**.



Polarization of Light

A vertical polarizer will pass only vertical components of the lights and horizontal components will be absorbed by it. Similarly a horizontal polarizer will pass only horizontal components and vertical components are absorbed.

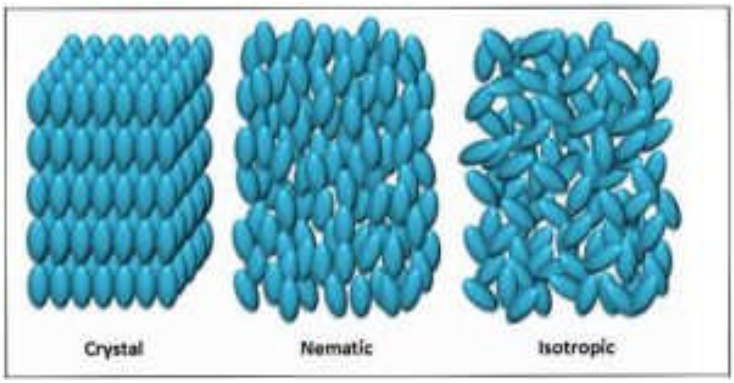


Vertical and Horizontal Polarizers

As shown in the above image we can block or pass light by changing the polarization.

Liquid Crystal

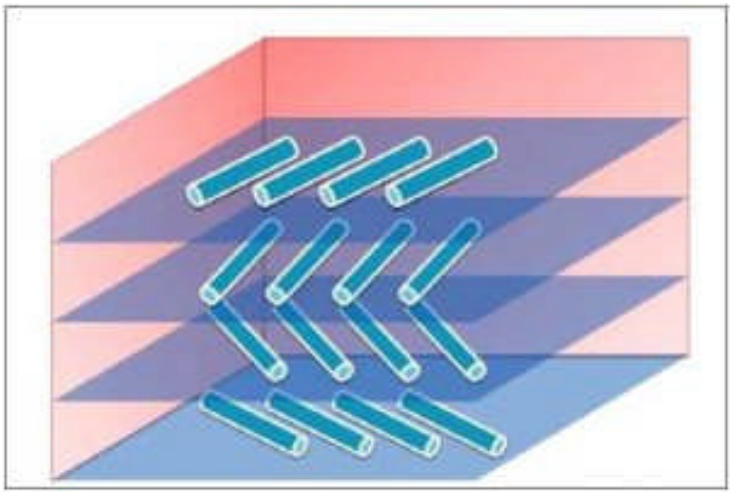
As the name indicates Liquid Crystals exists in a state between crystalline (solid) and isotropic (liquid) state. Among many phases, Nematic is a simplest form of liquid crystal phase which is employed in LCD technology.



Liquid Crystal Phases

Molecules of liquid crystal are long and cylindrical in shape. Each molecules in a plane are arranged in such a way that the major axis of each molecules are parallel

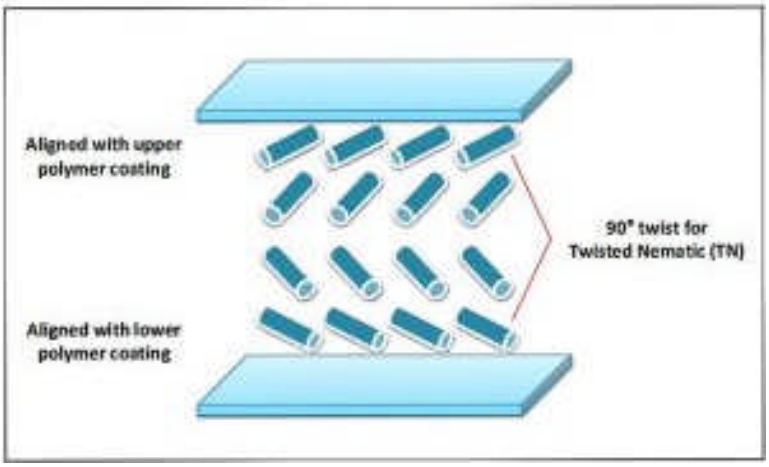
to each other. Orientation of molecules in each plane will be slightly different from the molecular orientation of adjacent planes as shown in the below diagram. This difference in molecular orientation in different planes will cause twisting the polarization of lights when it passes through it.



Orientation of Molecules – Liquid Crystal

Liquid crystals are affected by electric field, when we apply a voltage it will react and change its arrangement. This unique behaviour of liquid crystals made the key to LCDs.

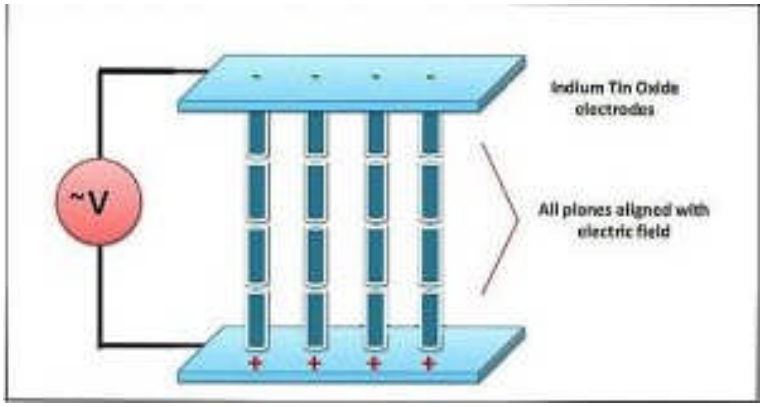
ON Pixel or Segment



Liquid Crystal – Orientation without Electric Field

In the above image we can see that molecules in each plane have different orientation without electric fields. So the polarisation of light changes when it passes through liquid crystal without electric field.

OFF Pixel or Segment

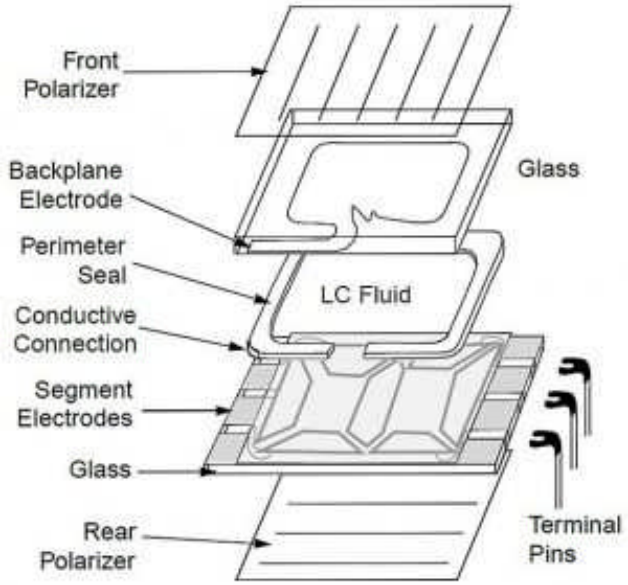


Liquid Crystal – Orientation with Electric Field

When an electric field is applied, we can see that all the molecules are arranged parallel to the same axis. So there will not be any change in the polarization of light when it passes through liquid crystal in an electric field.

Basic Components of LCD Panel

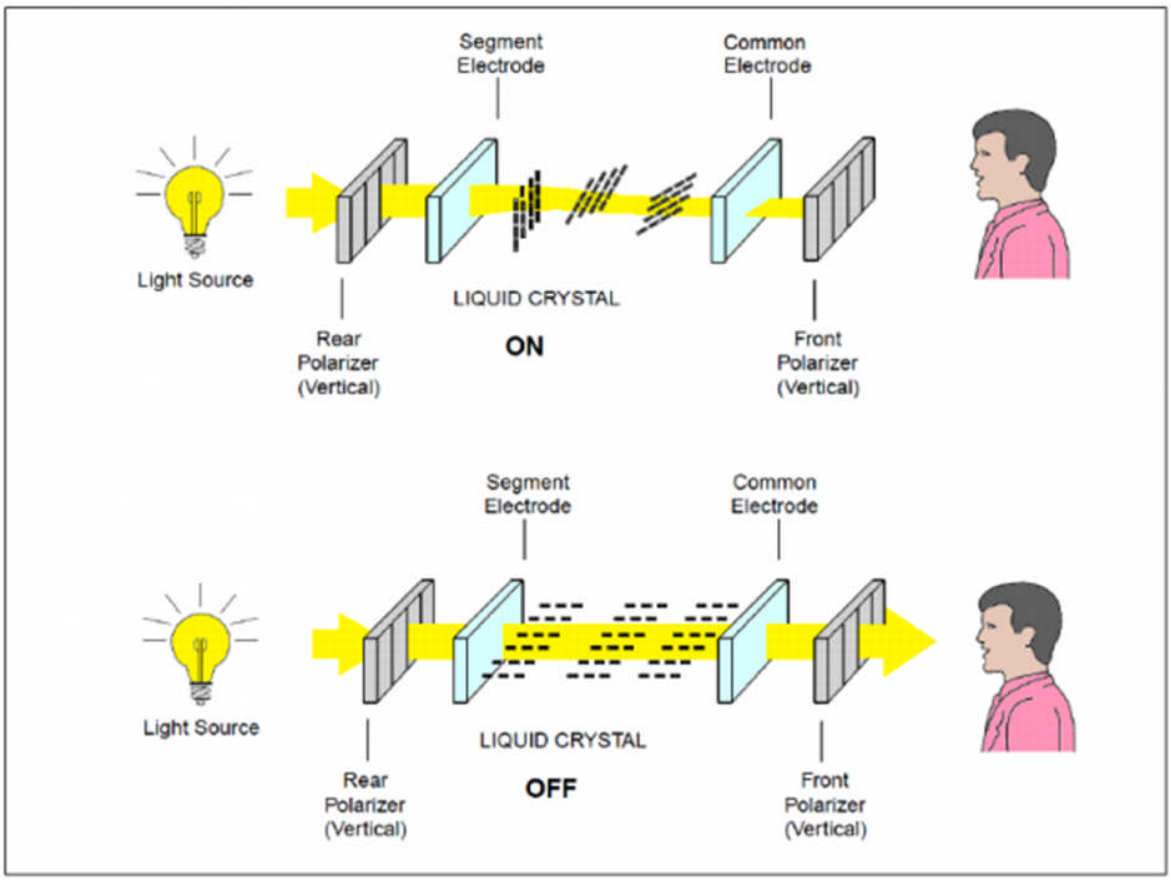
LCD panel looks like a piece of glass and it is commonly called as LCD Glass (displays used in calculators). It is constructed of different layers as shown in the following diagram.



Basic Components of LCD Display

As explained above, the twist in light polarization created by the liquid crystal is the basis of LCD operation. Now let's see the detailed working. There are basically two types of LCD displays, Transmissive and Reflective.

Transmissive Displays



Transmissive LCD Display Working

From above image, can easily understand the working of transmissive LCD display segment. At the left side we can see a light source which is emitting unpolarized light. When it passes through the rear polarizer (say vertical polarizer), the light will become vertically polarized. Then this light enters to the liquid crystal. As we seen before, liquid crystal will twist the polarization if it is ON. So when the vertically polarized light passes through ON liquid crystal segment, it becomes horizontally polarized. Next is front polarizer (say vertical polarizer), which will block horizontally polarized light. So that segment will appear as dark for the observer. If the liquid crystal segment is OFF, it will not change the polarization of light, so it will remain vertically polarized. So the front polarizer will pass that light. So it will appear as bright (not dark) for the observer.

This displays allows the use of back lights, commonly known as Backlit LCDs. We can also use ambient light as the source as used in the device shown below.

Transmissive LCD Display – Clock

Interfacing LCD with GrovePI & RaspberryPI

Steps-

1. Connect LCD display to any of I2C ports of GrovePI Hat
2. Run below code to change the color of display

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/12_LCD_Interfacing.py

Code-

```
from grove_rgb_lcd import *
import time
import random
while True:
    p1=random.randint(0,255)
    p2=random.randint(0,255)
    p3=random.randint(0,255)
    time.sleep(0.5)
    setRGB(p1,p2,p3) # (Green) RGB Pattern

    setText("Hi Welocme to Code Unnati")
```

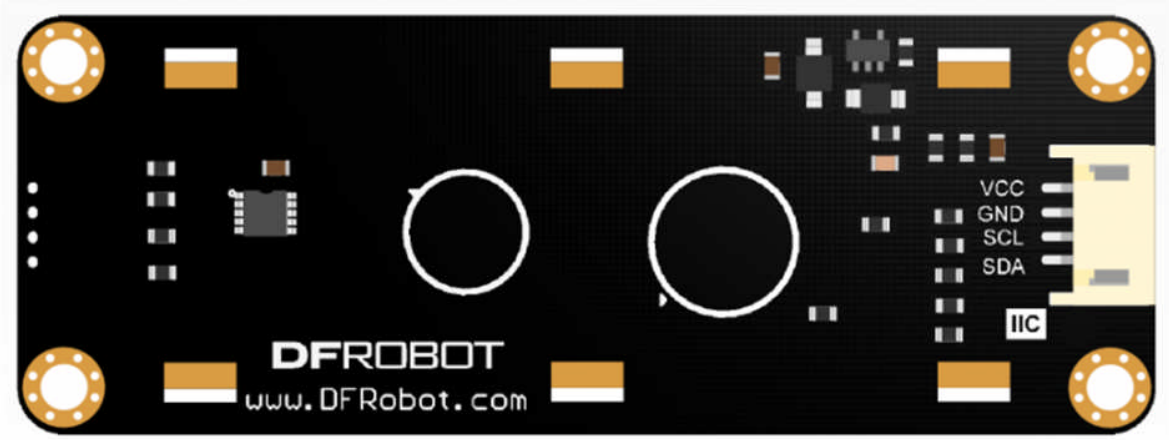
Interfacing LCD with DFRobot Hat & RaspberryPI

DFRobot kit provides a great I2C 16x2 LCD display compatible with Gadgeteer modules from DFRobot. With limited pin resources, your project will quicly run out of resources using normal LCDs. With this I2C interface LCD module, you only need 2 lines (I2C)to display the information.

Specification

- Operating Voltage: 3.3V~5.0V
- Operating Current: ≤60mA
- Display Description: 16*2
- Communication Mode: IIC/I2C
- Backlight: RGB can adjust backlight
- OperatingTemperature: -20 to 70℃
- Storage Temperature: -30 to 80℃
- Dimension: 87.0*32.0*13.0mm/3.43*1.26*0.51in

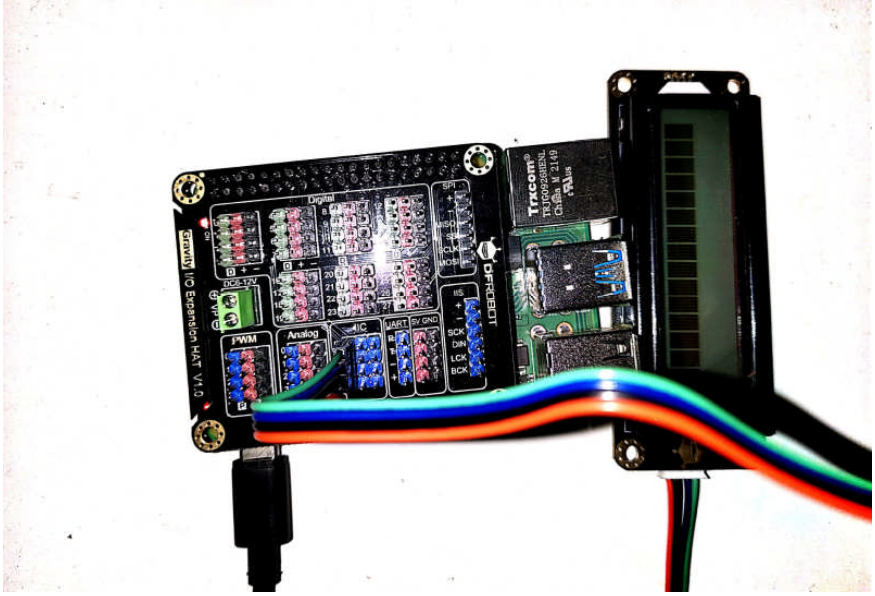
Board Overview



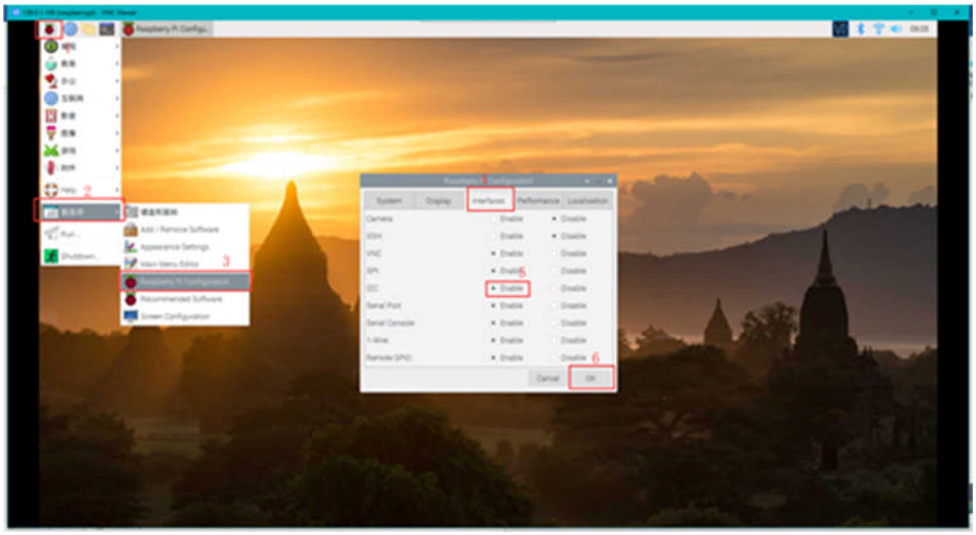
DFR LCD Unit

Use I2C LCD on Your Raspberry Pi

- Power the Raspberry Pi on and install the Raspberry Pi expansion board correctly
- Connect the sensor to corresponding IIC interface on the expansion board

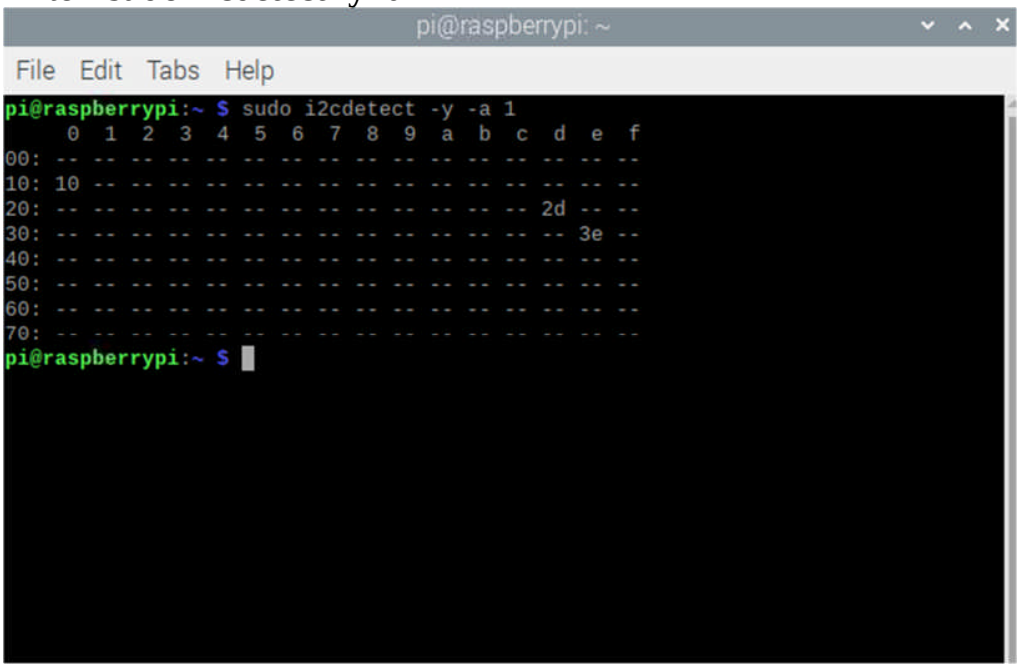


- Configure to enable I2C and restart the Raspberry Pi. If configured, you can skip this step. Configure the Raspberry Pi according to the following procedure and restart it.
-

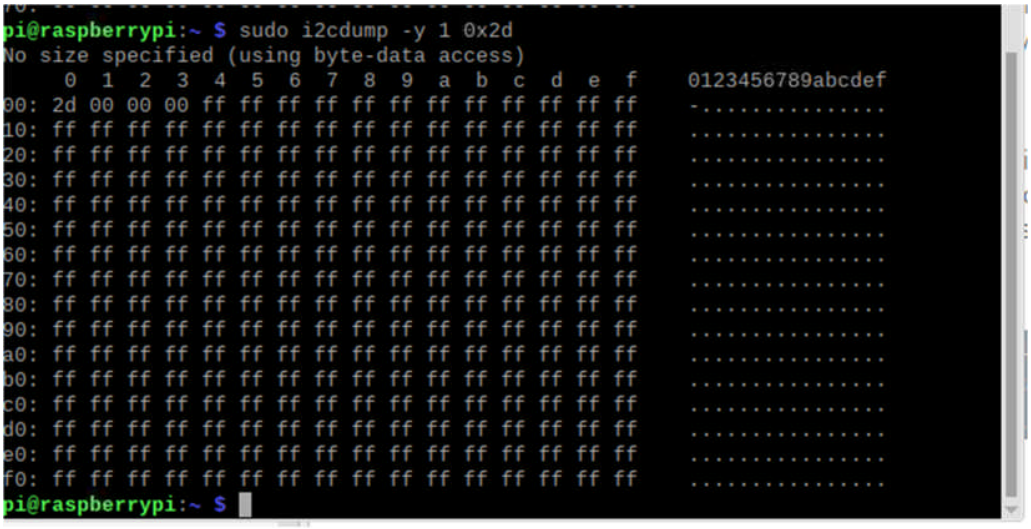


-

- When the I2C device is connected, the I2C address can be checked by the following command. In the terminal, type the following instructions and press
- 'Enter' **sudo i2cdetect -y -a 1**



- Read all register data of I2C device. In the terminal, type the following instructions and press 'Enter' sudo i2cdump -y 1 0x2d
-



- y means cancelling the user interaction process and directly executing the command 1 is the I2C device number 0×2d is I2C device address
- Open Thonny Python IDE to copy the following program into it

- Save below code (library of lcd functions) in a file *"lcddf.py"*

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/lcddf.py

Code-

```
import time,sys

if sys.platform == 'uwp':
    import winrt_smbus as smbus
    bus = smbus.SMBus(1)
else:
    import smbus
    import RPi.GPIO as GPIO
    rev = GPIO.RPI_REVISION
    if rev == 2 or rev == 3:
        bus = smbus.SMBus(1)
    else:
        bus = smbus.SMBus(0)

# this device has two I2C addresses
DISPLAY_RGB_ADDR = 0x2d
DISPLAY_TEXT_ADDR = 0x3e

# set backlight to (R,G,B) (values from 0..255 for each)
def setRGB(r,g,b):
    bus.write_byte_data(DISPLAY_RGB_ADDR,0,0)
    bus.write_byte_data(DISPLAY_RGB_ADDR,1,0)
    bus.write_byte_data(DISPLAY_RGB_ADDR,0x08,0xaa)
    bus.write_byte_data(DISPLAY_RGB_ADDR,4,r)
    bus.write_byte_data(DISPLAY_RGB_ADDR,3,g)
    bus.write_byte_data(DISPLAY_RGB_ADDR,2,b)

# send command to display (no need for external use)
def textCommand(cmd):
    bus.write_byte_data(DISPLAY_TEXT_ADDR,0x80,cmd)

# set display text \n for second line(or auto wrap)
```

```
def setText(text):
    textCommand(0x01) # clear display
    time.sleep(.05)
    textCommand(0x08 | 0x04) # display on, no cursor
    textCommand(0x28) # 2 lines
    time.sleep(.05)
    count = 0
    row = 0
    for c in text:
        if c == '\n' or count == 16:
            count = 0
            row += 1
            if row == 2:
                break
            textCommand(0xc0)
            if c == '\n':
                continue
        count += 1
        bus.write_byte_data(DISPLAY_TEXT_ADDR,0x40,ord(c))

#Update the display without erasing the display
def setText_norefresh(text):
    textCommand(0x02) # return home
    time.sleep(.05)
    textCommand(0x08 | 0x04) # display on, no cursor
    textCommand(0x28) # 2 lines
    time.sleep(.05)
    count = 0
    row = 0
    while len(text) < 32: #clears the rest of the screen
        text += ' '
    for c in text:
        if c == '\n' or count == 16:
            count = 0
            row += 1
            if row == 2:
                break
            textCommand(0xc0)
            if c == '\n':
                continue
```

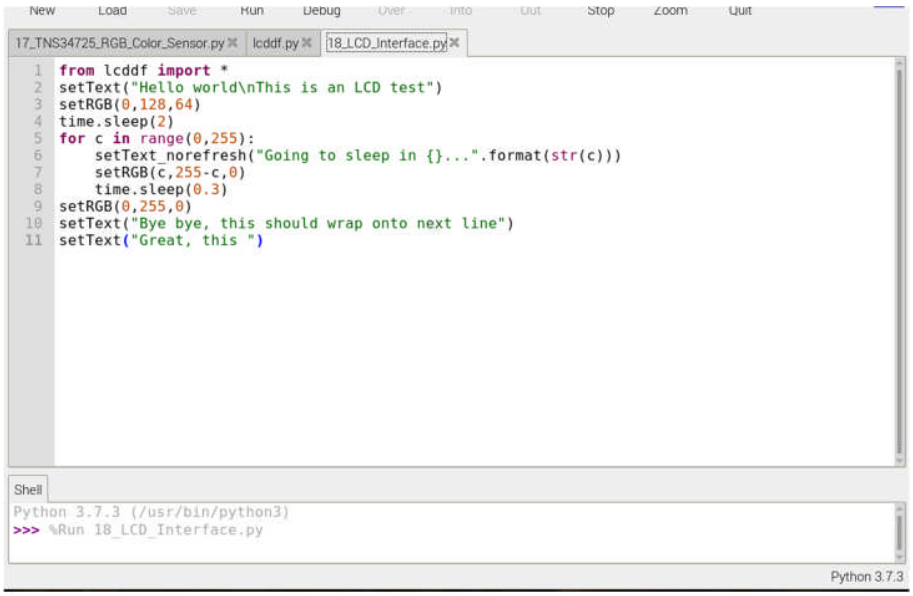
```
        count += 1
        bus.write_byte_data(DISPLAY_TEXT_ADDR,0x40,ord(c))

# Create a custom character (from array of row patterns)
def create_char(location, pattern):
    """
    Writes a bit pattern to LCD CGRAM

    Arguments:
    location -- integer, one of 8 slots (0-7)
    pattern -- byte array containing the bit pattern, like as found at
            https://omerk.github.io/lcdchargen/
    """
    location &= 0x07 # Make sure location is 0-7
    textCommand(0x40 | (location << 3))
    bus.write_i2c_block_data(DISPLAY_TEXT_ADDR, 0x40, pattern)
```

- Now let's load the library and create a neat script to interface with LCD

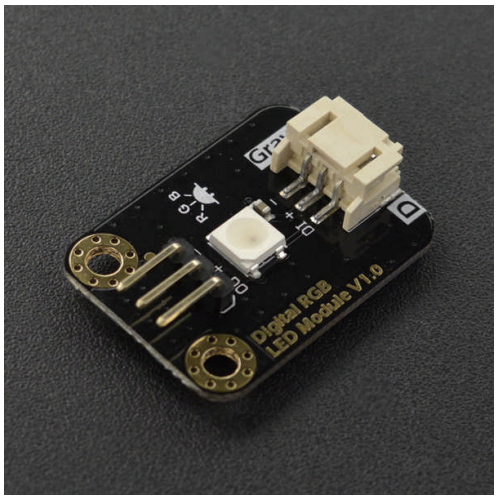
Github Link
https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/19_DFR_I2C_LCD_Interface.py

Code-

```
from lcddf import *
setText("Hello world\nThis is an LCD test")
setRGB(0,128,64)
time.sleep(2)
for c in range(0,255):
    setText_norefresh("Going to sleep in {}...".format(str(c)))
    setRGB(c,255-c,0)
    time.sleep(0.3)
setRGB(0,255,0)
setText("Bye bye, this should wrap onto next line")
setText("Great, this ")
```



RGB Led Interface



RGB Led Sensor

Digital RGB LED Module is a cascadable single RGB LED module compatible with RGB LED strip. The Gravity interface provides a more elegant manner for the con-nections between modules and most of mainstream controllers such as Raspberry Pi with an abundant kinds of DFRobot IO expansion shield. Compared to the tradi-tional RGB LED module, where three control signal pins are required for a single LED, this module only needs one signal pin for all LEDs in cascade.

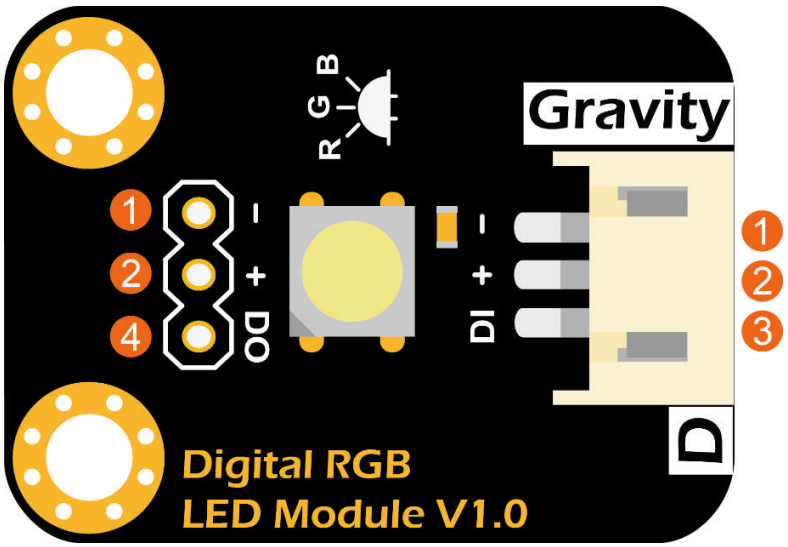
Thanks to such individual module design, RGB LED strip built by such independent modules can re-alize extremely low power consumption per meter compared to traditional RGB LED strip. And such feature makes it possible to cascade more than 100 modules with a total length up to several tens of meters but power requirement no more than 5V 2A. This also makes it much simpler and more flexible to configure the length of an RGB LED strip, without the need of cutting a RGB LED strip or soldering every LED pieces.

The module adopts the same high-brightness RGB LED in the traditional RGB LED strip, inheriting its excellent display performance and is fully compatible with the driving circuit or program.

Specification

- LED Type: WS2812 RGB 5050 LED
- Input Voltage: 3.3V~5.5V
- Maximum Current: 48mA
- Quiescent Current: 0.7mA
- Interface: Gravity 3P Digital
- Dimension: 30.0mm*22.0mm / 1.18*0.87 in

Board Overview



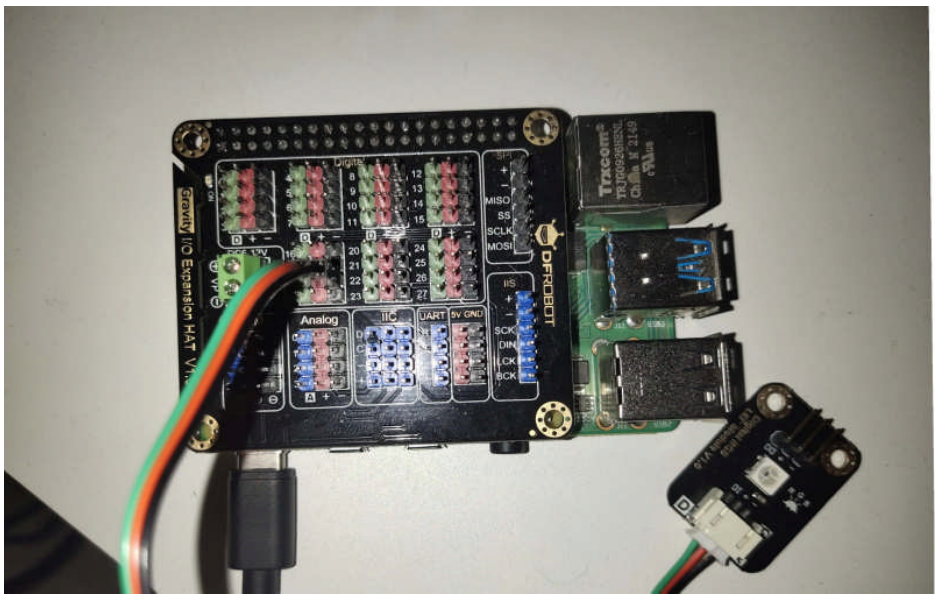
RGB Led Board

No.	Label	Description
1	-	Power supply GND
2	+	Power supply VCC（3.3~5.3V）
3	DI	Control data signal input
4	DO	Control data signal output

Interfacing RGB Led on DFRobot Hat & RaspberryPI

Steps-

1. Power the Raspberry Pi on and install the Raspberry Pi expansion board correctly
2. Connect the sensor to corresponding Digital port no. 18 interface on the expansion board



3. We need to install third party Adafruit tools and libraries to interface WS2812
4. Enter below commands sequentially in terminal

sudo pip3 install rpi_ws281x

sudo pip3 install adafruit-circuitpython-neopixel

```
sudo python3 -m pip install --force-reinstall adafruit-blinka
```

5. Open Thonny Python IDE to copy the following program into it

Github Link
https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/20_DFR_RGB_led.py

Code-

```
#sudo pip3 install rpi_ws281x
#sudo pip3 install adafruit-circuitpython-neopixel
#sudo python3 -m pip install --force-reinstall adafruit-blinka
#include all neccessary packages to get LEDs to work with Raspberry Pi
import time
import board
import neopixel
import random


#Initialise a strips variable, provide the GPIO Data Pin
#utilised and the amount of LED Nodes on strip and brightness (0 to 1 value)
pixels1 = neopixel.NeoPixel(board.D18, 1, brightness=0.5)

#Also create an arbitary count variable
x=0
while True:
    p1=random.randint(0,255)
    p2=random.randint(0,255)
    p3=random.randint(0,255)
    pixels1.fill((p1, p2, p3))
    time.sleep(0.5)

#Add a brief time delay to appreciate what has happened
time.sleep(4)

#Complete the script by returning all the LED to off
pixels1.fill((0, 0, 0))
```

- Save the above code in a file “RGB_led_test.py”

- Run the program from terminal to control the RGB led as,

sudo python3 RGB_led_test.py