# Program to test 16x2 LCD

# Connect from CN9 to CN31 and c10 to c29

#Before Executing the program make sure that pin number 7 of raspberry pi is enabled as GPIO Pin

#open cofig file using command

#sudo nano /boot/config.txt

#Disable the follwing line by adding # infront of the line

#dtoverlay = w1-gpio

#import

import RPi.GPIO as GPIO

import time

import Adafruit\_DHT

sensor = Adafruit\_DHT.DHT11

pin = 4

light = 37

# Define GPIO to LCD mapping

LCD\_RS = 35

LCD\_E = 33

LCD\_D4 = 23

LCD\_D5 = 21

LCD\_D6 = 19

LCD\_D7 = 24

# 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.BOARD) # 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(light,GPIO.IN)

# Initialise display

lcd\_init()

while True:

humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

print ("Humidity ="+str(humidity))

lcd\_string('H=%s '%humidity+'&'+' T=%s'%temperature,LCD\_LINE\_1)

print ("Temperature ="+str(temperature))

#lcd\_string('T=%s'%temperature,LCD\_LINE\_1)

if GPIO.input(light):

print("Light not Detected")

lcd\_string("Light not Detected",LCD\_LINE\_2)

while GPIO.input(light):

time.sleep(0.3)

#lcd\_byte(0x01,LCD\_CMD)

else: lcd\_string("Light Detected",LCD\_LINE\_2)

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()