Ex No:

DEVELOP AN APPLICATION TO CONTROL LIGHT BASED ON INTENSITY OF SUNLIGHT

Date :

AIM:

To Develop an IoT-based electronic device that can automatically control the light based on the intensity of light through switching On/OFF the street lights using Raspberry pi.

COMPONENTSREQUIRED:

COMPONENTS	NOS
RASPBERRY PI 3	1
LAMP	1
LM016L	1
MCP3208	1
RELAY	1
RESISTOR	1
TORCH_LDR	1

PROCEDURE:

Step1:Open proteus8 IDE,file->new project.

Step2:Select the create firmware project and go to the family and click on raspberry pi.

Step3: Select the lcd display,Torch ,LDR ,Ground,Default(terminal),MCP3208,Relay.

Step4: Placeallthe componentsin theworkspace.

Step5: Connect the LCD Display 5th Pin to the Ground, 4th Pin to GPI04, 6th Pin to GPIO70 and also (11^{th},12^{th},13^{th},15) To (GPIO18,GPIO27,GPIO22,GPIO23)Raspberry pi.

Step6:Connect the MCP3028 (13,11,12,10)pins to (CLK,MOSI,MISO,CS)raspberry and 14th pin toground.

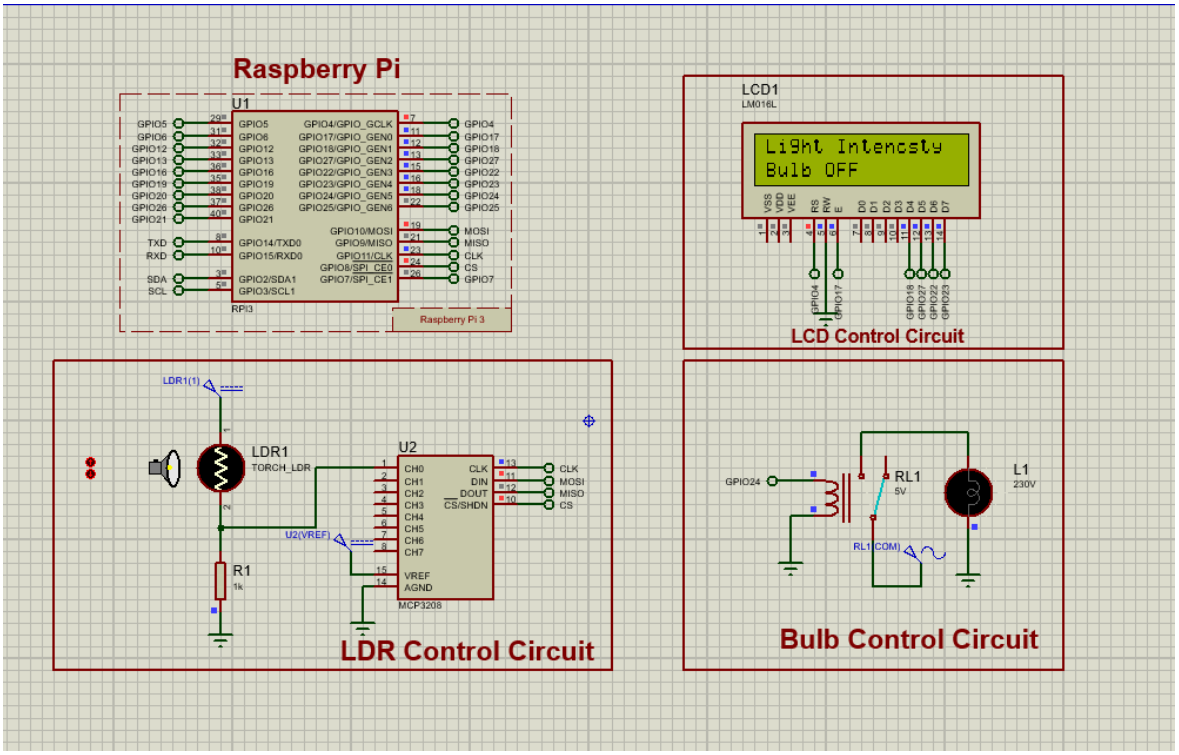
Step 7:Go to toolbar select DC under the generator mode and connect the DC to 15th pin of MCP3208

Step8:Select torch_LDR and connect them to DC and ground ,connect ground to 1st pin of MCP3208.

Step 9: Select the 5vlt relay connect them to GPI024 andground

Step10:Select the lightbulb and connect them to relay andground

SCHEMATIC DIAGRAM:



PROGRAM:

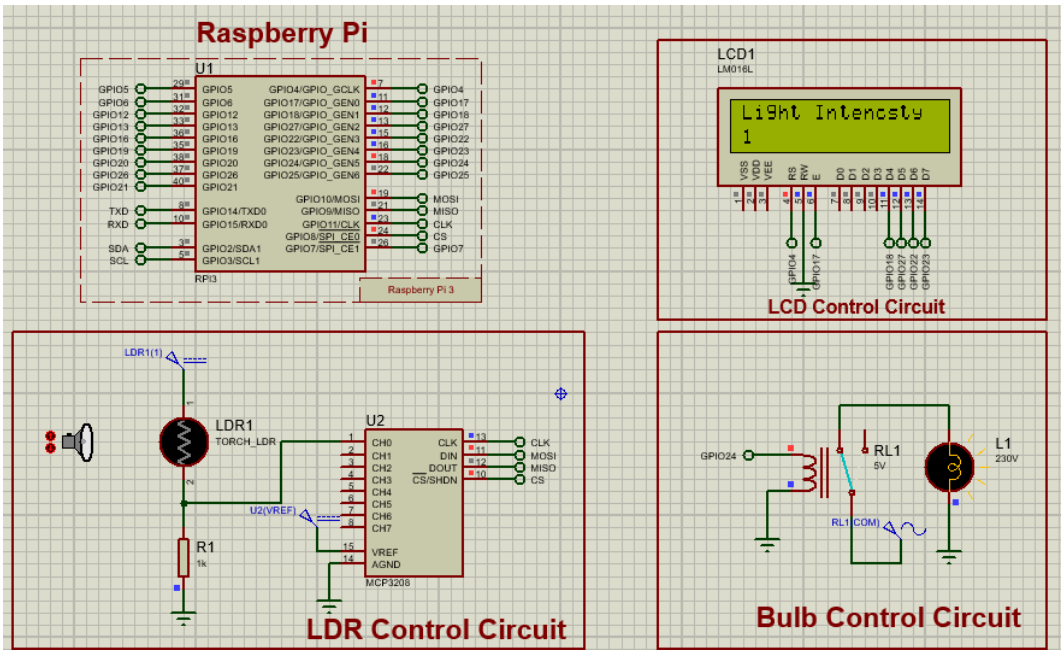
```
#!/usr/bin/python
import spidev
import time
import os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)
# Define GPIO to LCD mapping
LCD_RS = 7
LCD_E  = 11
LCD_D4 = 12
LCD_D5 = 13
LCD_D6 = 15
LCD_D7 = 16bulb_pin =  18
#Define sensor channels
temp_channel  = 0
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1
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(bulb_pin, GPIO.OUT) # DB7
```

```
# 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
                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)
```

```
# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
adc = spi.xfer2([1,(8+channel)<<4,0])
data = ((adc[1]&3) << 8) + adc[2]
return data
# Define delay between readings
delay = 5
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(1)
while 1:
light_level = ReadChannel(temp_channel)
# Print out results
lcd_string("Light Intencsty  ",LCD_LINE_1)
lcd_string(str(light_level),LCD_LINE_2)
time.sleep(1)
if(light_level< 100 ):
lcd_string("Bulb ON",LCD_LINE_2)
GPIO.output(bulb_pin, True)
time.sleep(1)
else:
lcd_string("Bulb OFF",LCD_LINE_2)
GPIO.output(bulb_pin, False)
time.sleep(1)
```

OUTPUT:



RESULT:

Thus the above program to simulate and control the light using raspberry in Proteus was executed and verified successfully.