San Francisco Bay University

CE450 Fundamentals of Embedded Engineering
Lab 3 RGB LEDs Control

Objectives:

In this lab, students will write Python programming to control RGB LEDs on Raspberry Pi board and do hands-on exercise through lab assignments.

Introduction:

RGB LEDs with three different colors can be controlled by Python program on Raspberry Pi board through GPIO. This lab is to create a design that causes LEDs on or off in the different manners

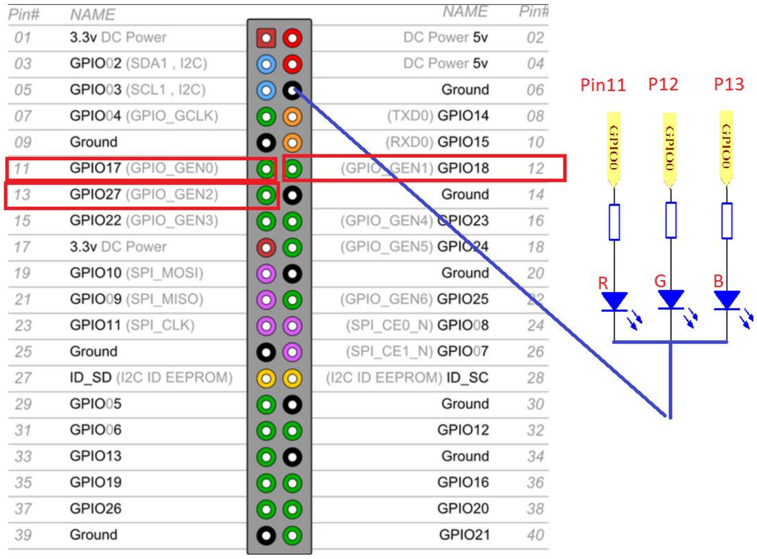
Equipment:

The equipment you require is as follows:

- Laptop & Raspberry Pi 4B model Board
- SunFounder Super Starter Kit V2.0 for Raspberry Pi

The Laboratory Procedure:

1. **Hardware connection**



Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

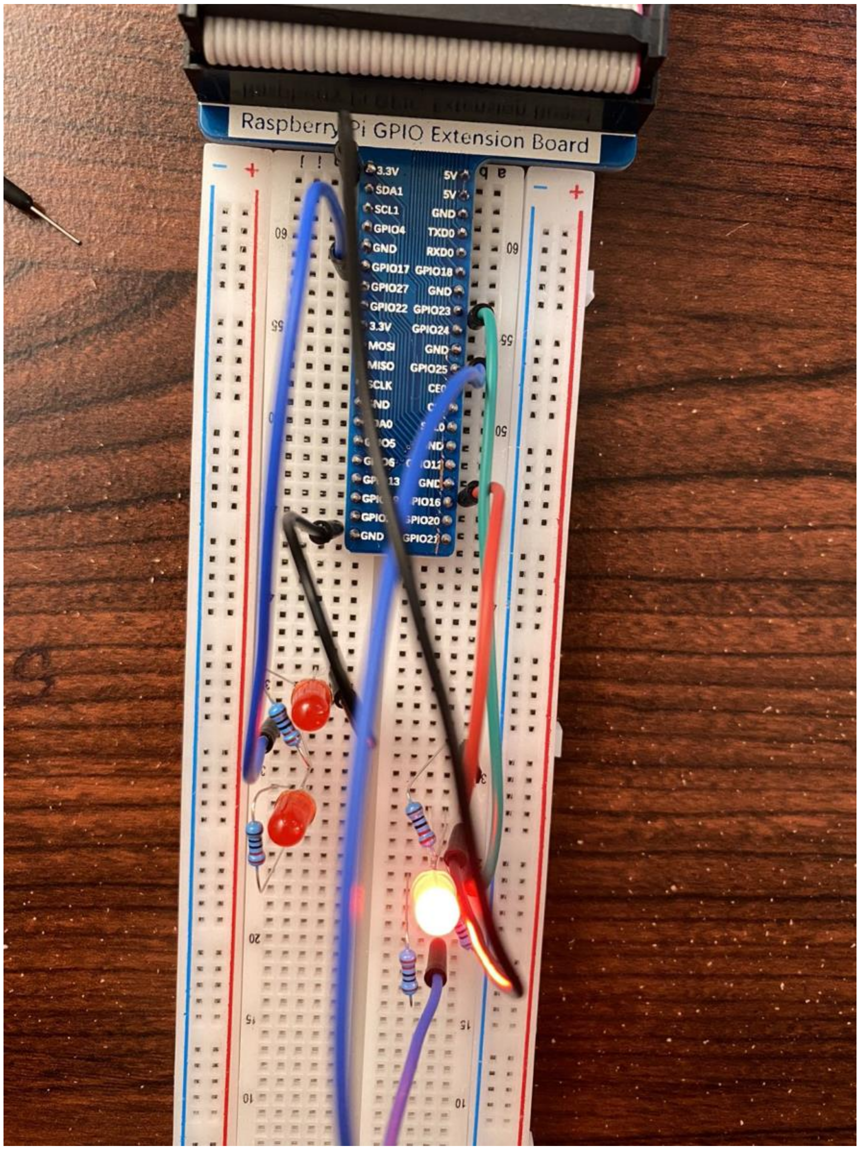
2. Control program in Python

```
# Python program

import RPi.GPIO as GPIO
import time

# R: GPIO16 pin#36, G: GPIO23 pin#16, B: GPIO25 pin#22, long leg to 3.3V power
colors = [0xFF0000, 0x00FF00, 0x0000FF, 0xFFFF00, 0xFF00FF, 0x00FFFF]
pins = {'pin_R':36, 'pin_G':16, 'pin_B':22}           # pins is a dict data type

GPIO.setmode(GPIO.BOARD)                              # Numbers GPIOs by physical
location
for i in pins:
         GPIO.setup(pins[i], GPIO.OUT)                # Set pins' mode is output
         GPIO.output(pins[i], GPIO.HIGH)              # Set pins to high(+3.3V) to
switch on led

p_R = GPIO.PWM(pins['pin_R'], 2000)                   # set Frequency to 2KHz
p_G = GPIO.PWM(pins['pin_G'], 2000)
p_B = GPIO.PWM(pins['pin_B'], 5000)

p_R.start(0)                                 # Initial duty Cycle = 0(leds off)
p_G.start(0)
p_B.start(0)

def map(x, in_min, in_max, out_min, out_max):
         return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def setColor(col):                                    # For example: col = 0x112233
         R_val = (col & 0xFF0000) >> 16
         G_val = (col & 0x00FF00) >> 8
         B_val = (col & 0x0000FF) >> 0

         R_val = map(R_val, 0, 255, 0, 100)
         G_val = map(G_val, 0, 255, 0, 100)
         B_val = map(B_val, 0, 255, 0, 100)

         p_R.ChangeDutyCycle(R_val)                   # Change duty cycle
         p_G.ChangeDutyCycle(G_val)
         p_B.ChangeDutyCycle(B_val)

try:
         while True:
                  for col in colors:
                           setColor(col)
                           time.sleep(0.5)
except   KeyboardInterrupt:
         p_R.stop()
         p_G.stop()
         p_B.stop()
         for i in pins:
                  GPIO.output(pins[i], GPIO.HIGH)                # Turn off all leds
         GPIO.cleanup()


#################################################################################

# Understanding the program
setColor(col = 0x FF  00  00)
                  R   G    B
```

```
R_val=0xFF0000 & 0xFF0000
     = 1111_1111_0 0 0 0 >> 16
     = 0 0 0 0_1111_1111 => 255
G_val = 0xFF0000 & & 0x00FF00 >> 8
      = 0x000000

B_val = (0xFF0000 & 0x0000FF) >> 0
      = 0x000000


(x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
e.g.      255 -> 100
          x   -> x*100/255

R_val = map(255, 0, 255, 0, 100)

                              100 - 0
      = (255-0) * ----------------- + 0
                              255 - 0
      = 100 => display Red color


setColor(col = 0x00FF00) =>
R_val = 0
G_val = 255       => Green
B_val = 0

setColor(col = 0x0000FF) =>
R_val = 0
G_val = 0
B_val = 255       => Blue

setColor(col = 0xFFFF00)
R_val = 255
G_val = 255
B_val = 0       final yellow color
https://www.w3schools.com/colors/colors_converter.asp

setColor(col = 0xFF00FF)
R_val = 255
G_val = 0
B_val = 255       final Fuchsia color

setColor(col =0x00FFFF)
R_val = 0
G_val = 255
B_val = 255       final Aqua color

#################################################################################
```

1. Reference Example Practice:

In my experiment, R is connected to GPIO16, G is connected to GPIO23, B is connected to GPIO25, long leg to 3.3v power.



Please click this [link](https://youtube.com/shorts/mej8Y3G1HK0?feature=share) for video.(https://youtube.com/shorts/mej8Y3G1HK0?feature=share)

2. The Laboratory Assignments:

1. International Morse code comprises the combinations of different dot and dish. In the following link is the encoding table for the alphabets and numbers *https://en.wikipedia.org/wiki/Morse_code*

If switching the red LED on as a dot, the blue with double on-time period of a dot as a dish and double on-time period of a dish as the time interval between two characters, please design Python program to send a message "TEST123" on RPi board periodically

My Answer:

Morse series for TEST123： (dish)-(dot)-(dot-dot-dot)-dish-(dot-dish-dish-dish-dish)-(dot-dot-dish-dish-dish)-(dot-dot-dot-dish-dish)

Python Code:

```
import RPi.GPIO as GPIO
import time

redLedPin = 31 # define redLedPin pin#31 <--> GPIO6
blueLedPin = 36 # define blueLedPin pin#36 <--> GPIO16
codeDict = {
'T' : ['dish'],
'E' : ['dot'],
'S' : ['dot', 'dot', 'dot'],
'1' : ['dot', 'dish', 'dish', 'dish', 'dish'],
'2' : ['dot', 'dot', 'dish', 'dish', 'dish'],
'3' : ['dot', 'dot', 'dot', 'dish', 'dish'],
'5' : ['dot', 'dot', 'dot', 'dot', 'dot']
}

def setup():
    GPIO.setmode(GPIO.BOARD) # use PHYSICAL GPIO Numbering
    GPIO.setwarnings(False)
    GPIO.setup(redLedPin, GPIO.OUT) # set the ledPin to OUTPUT mode
    GPIO.output(redLedPin, GPIO.LOW) # make ledPin output LOW level
    GPIO.setup(blueLedPin, GPIO.OUT) # set the ledPin to OUTPUT mode
   GPIO.output(blueLedPin, GPIO.LOW) # make ledPin output LOW level


def destroy():
    GPIO.cleanup() # Release all GPIO

def switchLed(ledPin, on):
    if on:
        GPIO.output(ledPin, GPIO.HIGH)
    else:
        GPIO.output(ledPin, GPIO.LOW)

def sleep(n_time_unit):
    for i in range(n_time_unit):
        time.sleep(0.3)
```

```
def showDot(n=1):
    for i in range(n):
        switchLed(redLedPin, True)
        sleep(1)
        switchLed(redLedPin, False)
        sleep(1)

def showDish(n=1):
    for i in range(n):
        switchLed(blueLedPin, True)
        sleep(2)
        switchLed(blueLedPin, False)
        sleep(1)

def showMorseCode(code):
    print("Morse code begins...")
    for c in code:
        for e in codeDict[c]:
            if e == 'dot':
                showDot()
            elif e == 'dish':
                showDish()
        sleep(4)
    print("Morse code ends...")

def my_loop(code):
    while True:
        showMorseCode(code)

if __name__ == '__main__': # Program entrance
    print ('Program is starting ... \n')
    setup()
    try:
        code = "TEST123"
        my_loop(code)
    except KeyboardInterrupt: # Press ctrl-c to end the program.
        destroy()
```

Practice-Video Link on Youtube:

https://youtu.be/6agdoM8VXfU