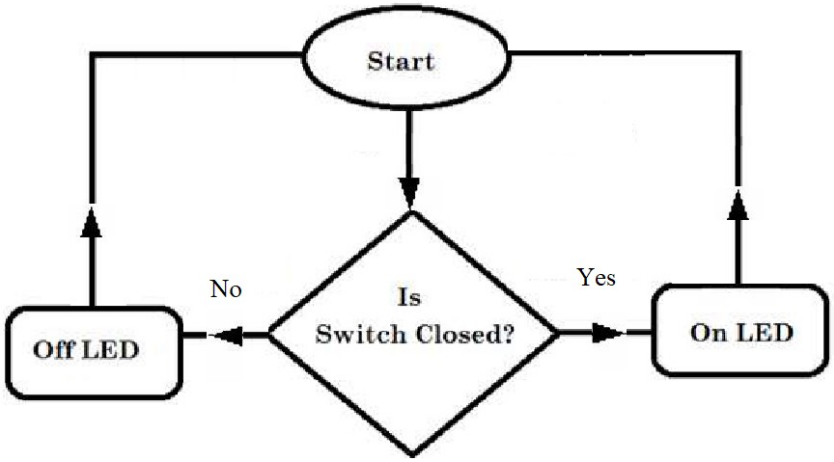
Lab Sheet: 2

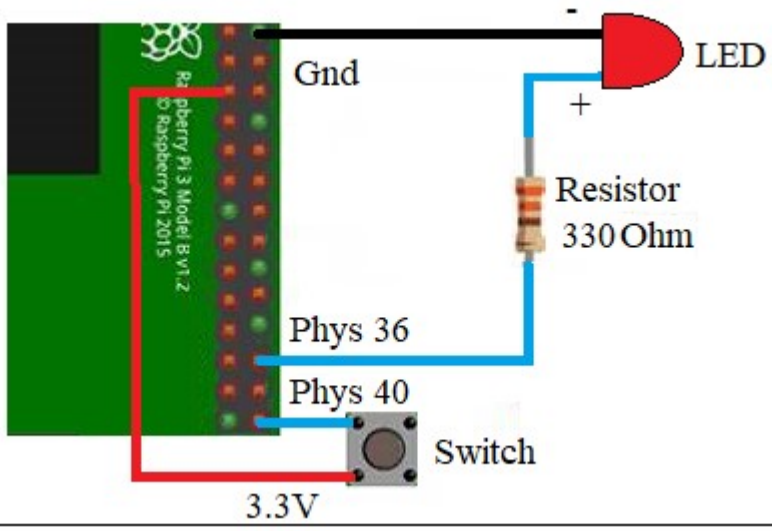
Objective: Interfacing simple input devices (switch/button) and 7-Segment with Pi

Hardware Requirements: Raspberry Pi; SD card at least 4GB, Monitor with HDMI (High definition multimedia interface) connector, Micro USB power supply, USB keyboard, USB mouse, breadboard, switches, 7-segments, resistors and connecting wires.

---

1. Interfacing a switch: write a code to control an LED with a switch; configure pin 36 as input and pin 40 as output (follow physical pin numbering schame); connect a switch to the input pin and an LED to the output; control the ON and OFF state of the LED through the switch.





Let us start with adding a push-button to the breadboard; connect the power rail of breadboard to a power pin (which is marked 3.3v) of the Raspberry Pi with a male-to-female jumper wire, then connect one leg of your push-button to the power rail with a male-to-male jumper wire; connect the other leg to your input pin 40 (i.e. GPIO21); make a connection to LED also; run the code and verify the output.

```
#Controlling an LED with a switch
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)        # following physical pin numbering
GPIO.setup(36, GPIO.OUT)        # pin 36 is configured as output
GPIO.setup(40, GPIO.IN, pull_up_down=GPIO.PUD_UP) # enable pull-up
# pin 40 is connected to input device -switch
while True:
        if GPIO.input(40)==True :    # if switch is pressed
             GPIO.output(36, True)   # then turn LED on
             print ("Button pressed")
        else :
            GPIO.output(36, False)   # else turn it off
            print ("Button not pressed")
GPIO.cleanup()
```

Will the following code work?
Justify your answer.

```
data=False
while True:
    data=GPIO.input(40)
    GPIO.output(36, data)
    time.sleep(1)
```

The above task is implemented with gpiozero library as follows

```
import RPi.GPIO as GPIO
from gpiozero import LED, Button
import time
GPIO.setwarnings(False)
led=LED(21)                    # BCM21=Phys40
button=Button(16)              # BCM16 =Phys36
while (True):
   if button.is_pressed:
      led.on()
   else:
      led.off()
GPIO.cleanup()
```

---

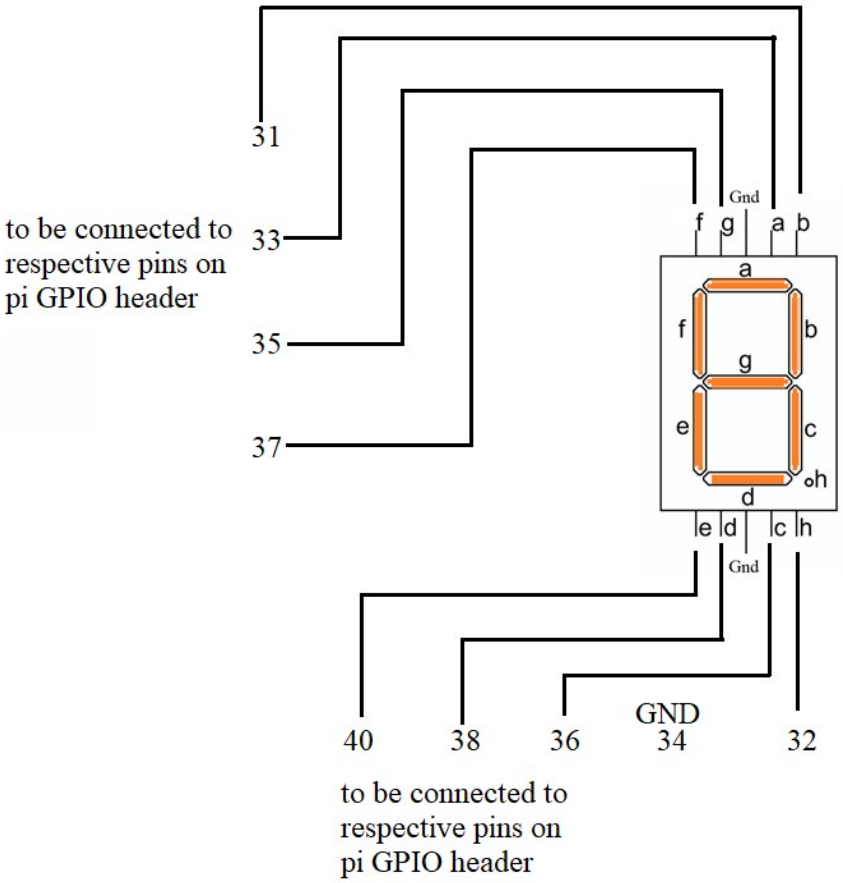
2. Modify the above code for operating two LEDs (L1 and L2) with two switches (S1 and S2)

3. Design a Running LED with an array of 7 LEDs; use Seven Segment Display (SSD); run the LED until 'n' to be pressed (reading input from keyboard)

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
pins =  [33,  31,  36,  38,  40,  37,  35]          # selecting pins for 7 segs
# seg =[a,     b,    c,    d,    e,    f,     g ]
# Physical pin 33 is connected to segment a
# and pin 35 is connected to segment g
```

```
for i in pins:                    # configuring 7 pins as output
   GPIO.setup(i,GPIO.OUT)
choice='n'
while choice!='n' :               # loop until choice not equal to 'n'
    for i in pins:                # since we use common cathode SSD
        GPIO.output(i,True)       # writing True wil lit LED
        time.sleep(1)             # sleep for a second
        GPIO.output(i,False)      # turn the LED off
    choice=input('Want to continue [y/n]? ') #read choice from keyboard

GPIO.cleanup()
```



4. Show the figure of eight and three on SSD one by one with a delay of two seconds; following schematic is constituted of SSD with common anodes; and the common pins are connected to the Gnd; so that, writing zero on a pin lit an LED.

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
pins = [33, 31, 36, 38, 40, 37, 35, 32]# selecting pins for 7 segs and dp
# Physical pin 33 is connected to segment-a of common cathode SSD
# and pin 35 is connected to segment-g

for i in pins:                          # configuring 7 pins as output
   GPIO.setup(i,GPIO.OUT)

for i in pins:                          # creating a figure of Eight
   GPIO.output(i,False)                 # since we use common anode SSD
                                        # writing False wil lit LEDs
time.sleep(2)                           # after 2 seconds show 3 on SSD
mask=0x01
for i in pins:                          # hex code for three: B0
   a=mask & 0xB0                        # extracting one bit of hex
   if a:                                # if  (a = =True)
      GPIO.output(i,True)               # write extracted bit on pin
   else:
      GPIO.output(i,False)
   mask=mask<<1                         # go to extract next bit of hex

GPIO.cleanup()
```

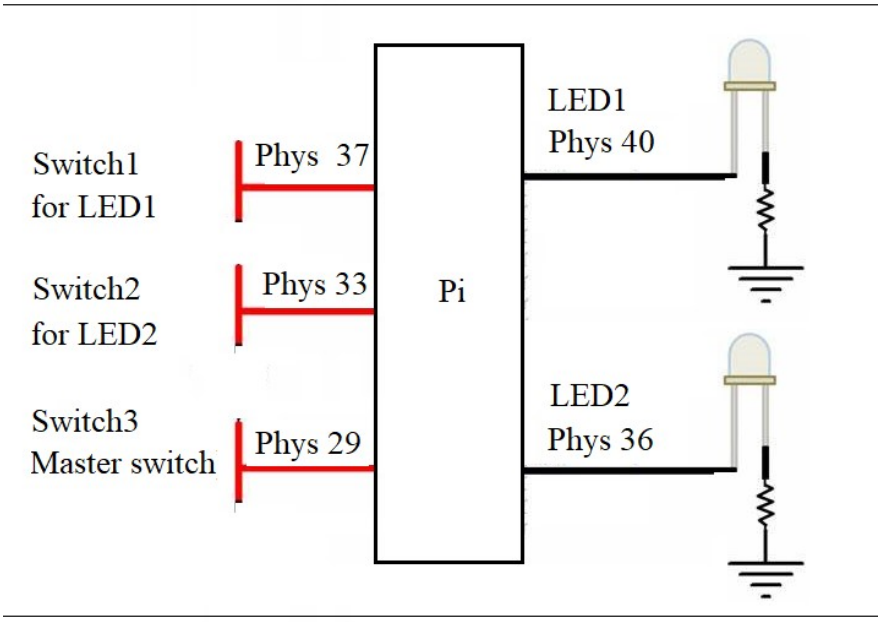
5. Design a decade counter which counts from 0 to 9; first step is to generate hexadecimal values for each Number or Character to be displayed on SSD; then write a code.

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
pins = [33, 31, 36, 38, 40, 37, 35, 32] # selecting pins for 7 segment and dp
pattern=[ 0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]
# Hence  we use common cathode SSD, writing True turns the LED on
for i in pins:                          # configuring the 7 pins as output
   GPIO.setup(i,GPIO.OUT)

## ……function definition…….
def ssdisplay (hex):
   mask=0x01
  for i in pins:
        a=mask & hex
        if a:
            GPIO.output(i, True)
        else:
            GPIO.output(i, False)
        mask=mask<<1
# ……function end…….
   for i in pattern:                    # send patterns 1 by 1 to SSD
        ssdisplay(i)                    # call function
        time.sleep(1)
GPIO.cleanup()
```

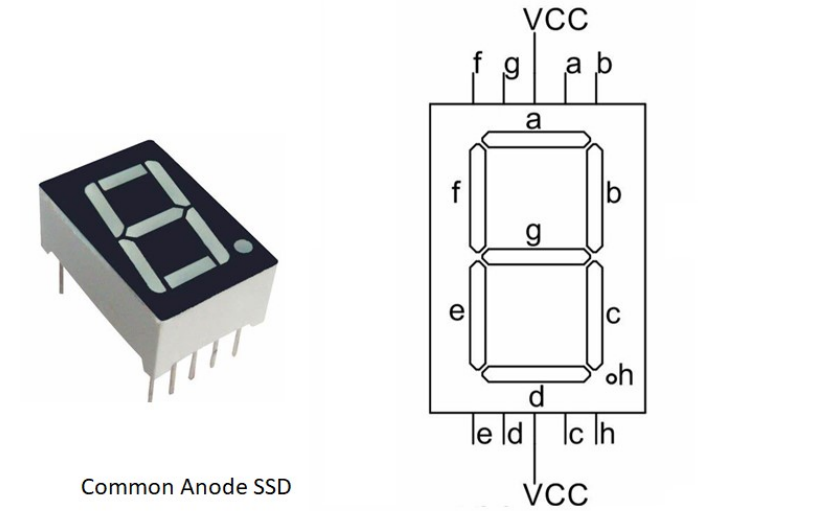
Assignment 1: Assume that there are two lights inside a hall, operated with two different switches; there is an additional master switch which is installed outside of the hall; LEDs can be operated through local switches only when the master switch is ON. Write a Phython code to implement the operation, use the following pseudo code.

```
If (switch3==1)     #checking master switch
      If (switch1==1) then LED1=1 else LED1=0
      If (switch2==1) then LED2=1 else LED2=0
Else
     LED1=0, LED2=0;
```



Assignment 2: Implement a single digit up-down counter with a switch; when the switch is off (by default) the counter become up-counter otherwise it become down-counter; use SSD.

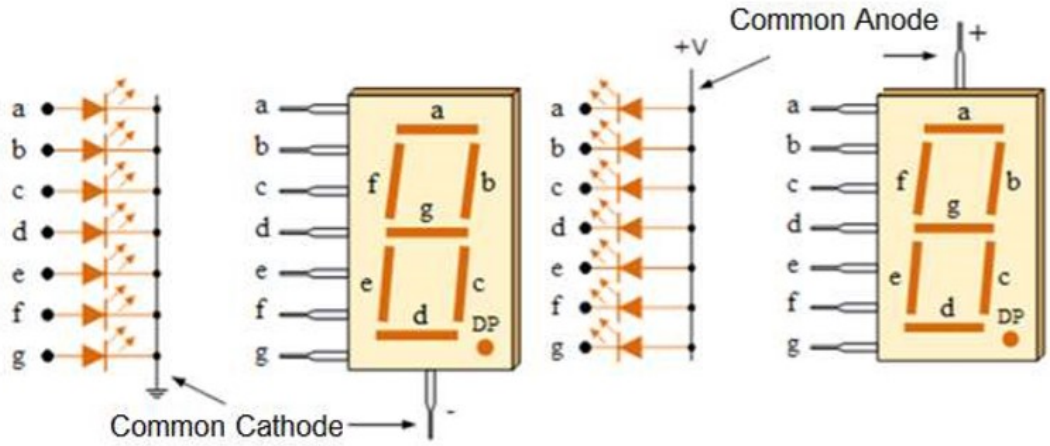
Seven Segment Display: A seven segment display (SSD) is the most basic electronic display device that can display digits from 0-9 and alphabets A to F. There is an array of seven LEDs arranged in a special pattern (hence its name) to display the digits and alphabets. SSD is generally available in as a ten pin package. While eight pins correspond to the seven LEDs, the two pins (at middle) are common and internally shorted. These segments come in two configurations, namely, Common cathode (CC) and Common anode (CA).



Common Anode SSD

Common Cathode: In the common cathode display, all the cathode connections of the LED segments are joined together to logic low (zero) or ground. The individual segments are illuminated by logic high (one) via a current limiting resistor to forward bias the individual Anode terminals (a-g). i.e any particular LED in the seven segment is glow(ON) by giving logic high voltage.

Common Anode: In the common anode display, all the anode connections of the LED segments are joined together to logic high. The individual segments are illuminated by applying a ground, logic low signal. For example, hex value 0x3F (011 1111) will create the figure of zero on a common cathode SSD; whereas in a common anode SSD, we need to use 0x40 (100 0000).



The following table shows hex codes for the digits zero to nine; note that the table assumes common anode SSD (Note that, if you change 1s into 0s and 0s into 1s, you will get hex code for common cathode SSD : 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F and 0x6F)

Digit	Seven Segment Conversion								Hex Number
	dot	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	1	0	0	0	98