LAB – 8 Programming on Interrupts

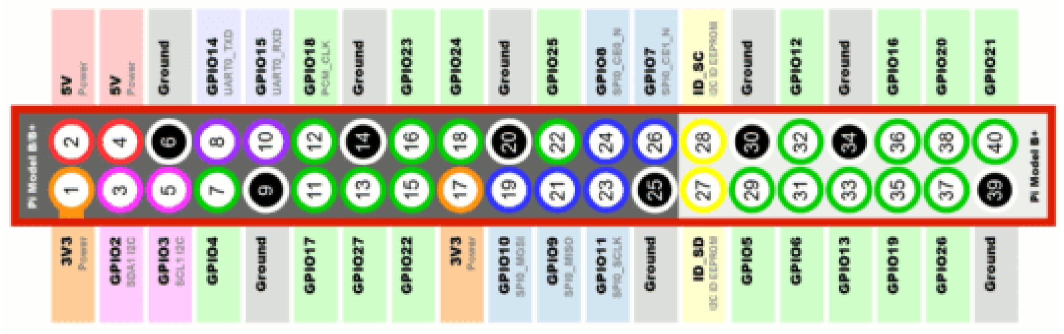
Aim:

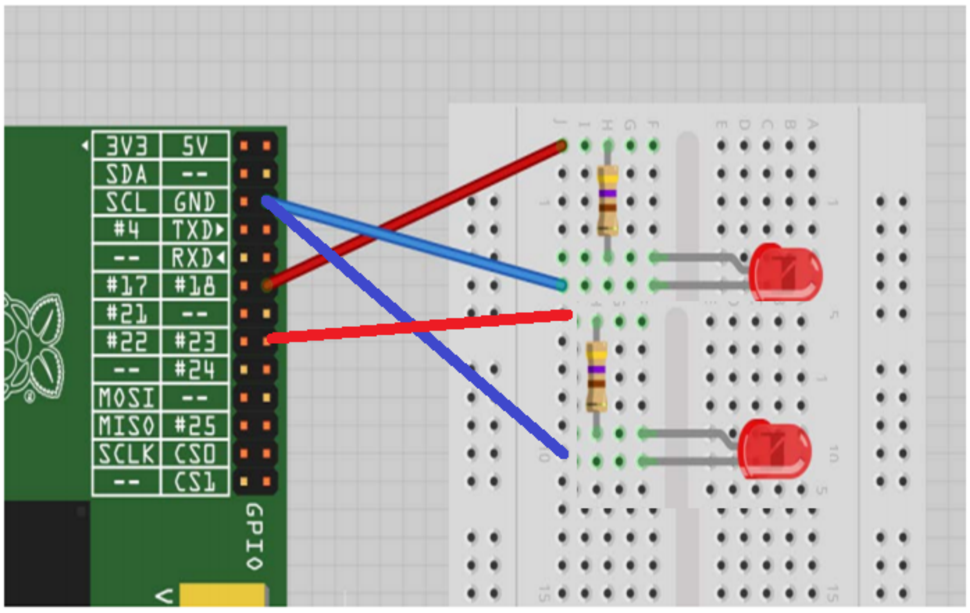
To write a program to switch on LEDs for a while, as a reaction to interrupts.

Task:

1. Power on the LED when the button is pressed, power off the LED when the button is released. Use inbuilt function GPIO.add_event_detect().
2. i)Configure two I/O pin for LOW to HIGH transition Interrupt source and HIGH to LOW transition Interrupt source
   ii)Write two Interrupt Service Routine(IRS) for LOW to HIGH transition Interrupt(ISRL2H) and HIGH to LOW transition Interrupt(ISRH2L), which will display(print)the occurrence of positive edge or negative edge Interrupt when it is called.
   iii)Make ON of LED1 **for a while**, if a positive edge interrupt signal occurs and call ISRL2H. Make ON of LED2 **for a while**, if the negative edge Interrupt signal occurs and call ISRH2L. Don't use the inbuilt function GPIO.add_event_detect() to detect interrupts.

Pin & Circuit Diagram:





Note: Interrupt sources are not shown in the figure.

Algorithm:

Task 1:

1. Import the necessary libraries: Import RPi.GPIO for GPIO control and the `sleep` function for delays.
2. Configure GPIO pins for buttons and an LED, suppressing warnings.
3. Define two callback functions:
   a.turn_on_led(channel): Set the LED pin HIGH and print a message when a button is pressed.
   b.turn_off_led(channel): Set the LED pin LOW and print a message when a button is released.
4. Add event detection for button presses and releases with debounce times.
5. Enter an infinite loop, keeping the program running.
6. Handle a keyboard interrupt with a try-except block, cleaning up GPIO on exit.

Task 2:

1. Import the RPi.GPIO library to control GPIO pins on a Raspberry Pi.
2. Set the GPIO mode to BCM.Configure GPIO pins 18 and 23 as output pins.
3. Define global variables to keep track of positive and negative edge occurrences.
4. Define two interrupt service routines (ISRs):
   a. ISRL2H(): Increment the positive edge count, print the count, set GPIO 18 to HIGH, wait for 2 seconds, set GPIO 18 to LOW, and wait for another 2 seconds.
   b. ISRH2L(): Increment the negative edge count, print the count, set GPIO 23 to HIGH, wait for 2 seconds (with an unnecessary extra delay), set GPIO 23 to LOW.

5.Prompt the user to input '1' for a positive edge or '0' for a negative edge.
6. Based on user input, call the appropriate ISR:
- If the user enters '1', execute **ISRL2H()** to simulate a positive edge.
- If the user enters '0', execute **ISRH2L()** to simulate a negative edge.

Programs:

Task 1:

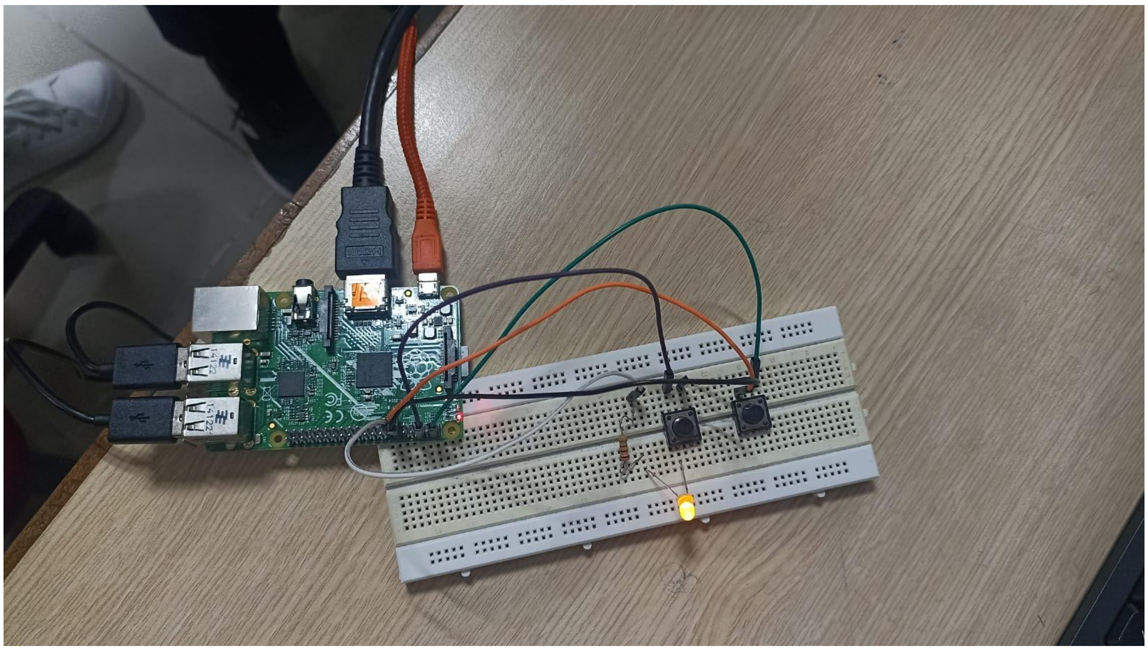
```python
import RPi.GPIO as GPIO
from time import sleep
button_pin1 = 17
button_pin2 = 18
led_pin =27
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(button_pin1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(button_pin2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led_pin, GPIO.OUT)

def turn_on_led (channel):
  GPIO.output (led_pin, GPIO.HIGH)
  print("switch 1 Rising, led ON")
def turn_off_led (channel):
  GPIO.output (led_pin, GPIO.LOW)
  print("switch 2 Falling, led OFF")
GPIO.add_event_detect (button_pin2, GPIO.FALLING, callback=turn_off_led, bouncetime=200)
GPIO.add_event_detect (button_pin1, GPIO.RISING, callback=turn_on_led, bouncetime=200)
print("in while loop")
try:
  while True:
       pass
except KeyboardInterrupt:
  GPIO.cleanup()
```

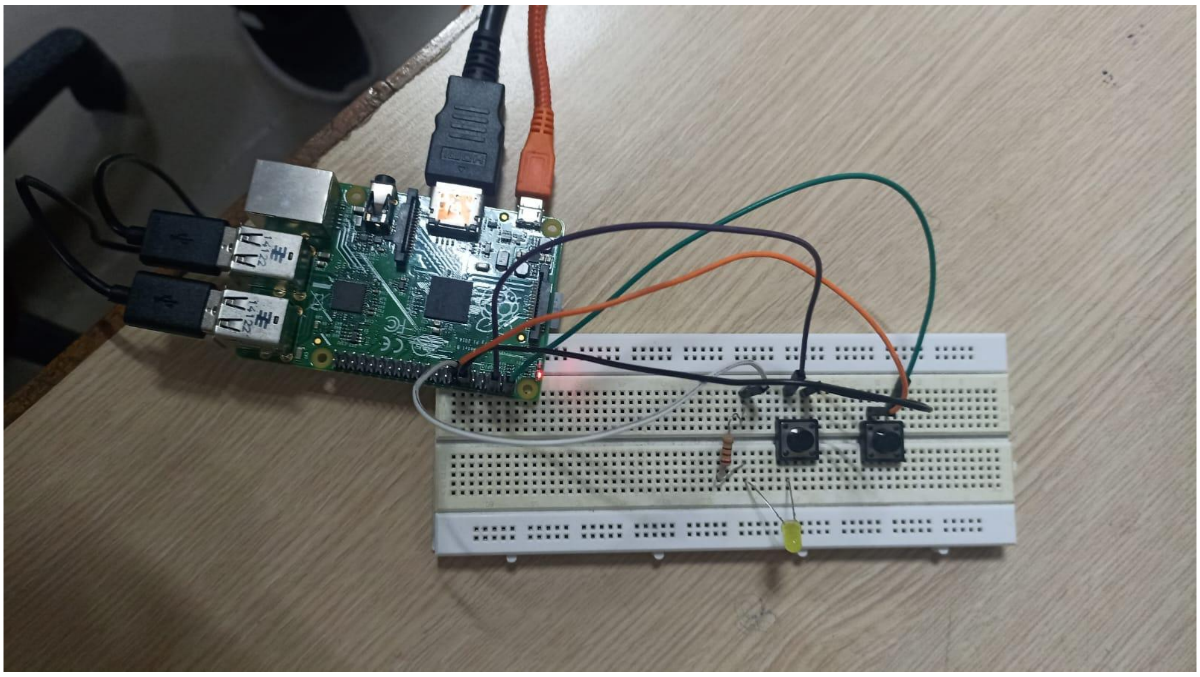
Task 2:

```python
import RPi.GPIO as GPIO
import time
GPIO.setmode (GPIO.BCM)
GPIO.setup(18,GPIO.OUT)
GPIO.setup(23,GPIO.OUT)
global positive_edge
positive_edge=0
global negative_edge
negative_edge=0

def ISRL2H():
    global positive_edge
    positive_edge+=1
    print("No of positive edge occurences:",positive_edge)
    GPIO.output(18,1)
    time.sleep(2)
    GPIO.output(18,0)
    time.sleep(2)
def ISRH2L():
    global negative_edge
    negative_edge+=1
    print("No of negative edge occurences:", negative_edge)
    GPIO.output(23,1)
    time.sleep(2)
    time.sleep(2)
    GPIO.output(23,0)

while True:
  GPIO.output(18,1)
  time.sleep(2)
  GPIO.output (23,1)
  GPIO.output(18,0)
  time.sleep(2)
  GPIO.output(23,0)
  time.sleep(2)
  ch=input("Enter 1 if there is a positive edge and 0 if there is a negative edge:")
  if ch=='1':
     ISRL2H()
  elif ch=='0':
     ISRH2L()
```

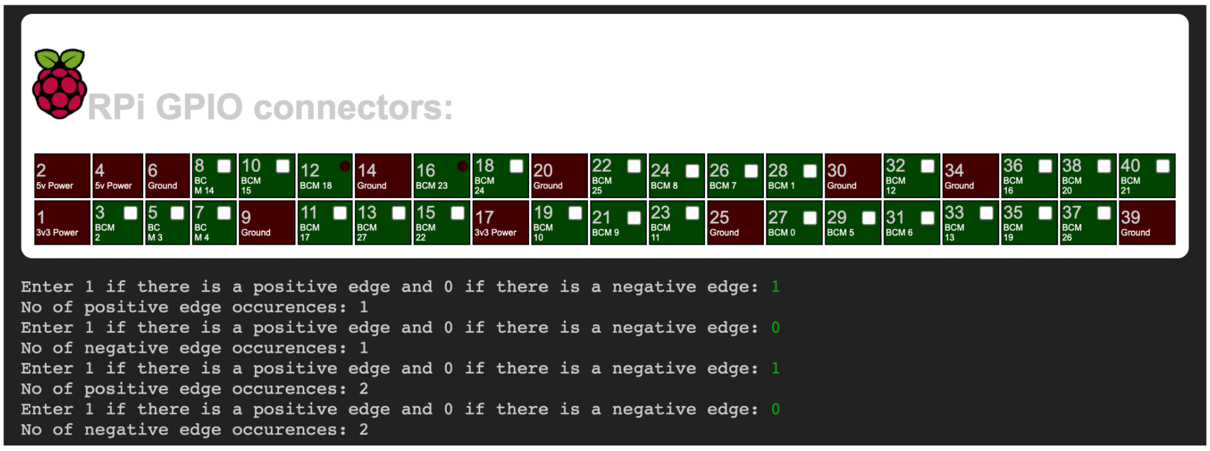
Output:

Task 1:





Task 2:



RPi GPIO connectors:

2 5v Power	4 5v Power	6 Ground	8 BC M 14	10 BCM 15	12 BCM 18	14 Ground	16 BCM 23	18 BCM 24	20 Ground	22 BCM 25	24 BCM 8	26 BCM 7	28 BCM 1	30 Ground	32 BCM 12	34 Ground	36 BCM 16	38 BCM 20	40 BCM 21
1 3v3 Power	3 BCM 2	5 BC M 3	7 BC M 4	9 Ground	11 BCM 17	13 BCM 27	15 BCM 22	17 3v3 Power	19 BCM 10	21 BCM 9	23 BCM 11	25 Ground	27 BCM 0	29 BCM 5	31 BCM 6	33 BCM 13	35 BCM 19	37 BCM 26	39 Ground

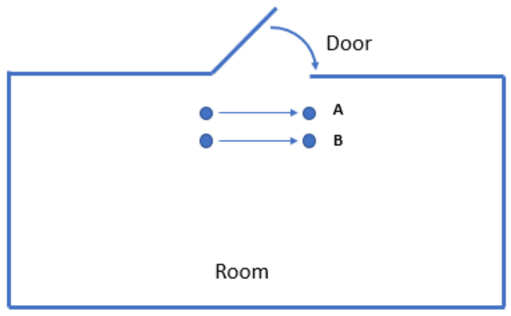
```
Enter 1 if there is a positive edge and 0 if there is a negative edge: 1
No of positive edge occurences: 1
Enter 1 if there is a positive edge and 0 if there is a negative edge: 0
No of negative edge occurences: 1
Enter 1 if there is a positive edge and 0 if there is a negative edge: 1
No of positive edge occurences: 2
Enter 1 if there is a positive edge and 0 if there is a negative edge: 0
No of negative edge occurences: 2
```

Pre-Lab Questions:

1. What is polling? Write the merit and demerit of polling?
2. Define interrupt? What do you understand about the interrupt service routine?
3. Explain the function ‘GPIO.add.event_detect( ).

Post Lab Questions:

1. In a room the number of persons entering and leaving has to be recorded automatically. A system is installed for this. The hardware contains two Infra-Red (IR) sources and a detector like our TV remote and TV system. Whenever IR light is broken a movement is detected. Two sensors are spaced apart to find out whether the person is leaving or entering. Sensor A is placed first and Sensor B is placed second. So, if Sensor B is triggered then Sensor A is triggered then it is treated as leaving. If it is Sensor A and B then it is treated as entering. For clarity please refer to the figure below.



Write a suitable pseudo code or algorithm for the above scenario. Explain the use of interrupts in this case.

2. Compare edge triggering and level triggering. Out of the two which one you will prefer. Justify your answer.

Result:

Thus,the python code to write a program to switch on LEDs for a while, as a reaction to interrupt in Raspberry Pi was written and successfully tested.