Laboratory Report Cover Sheet

SRM Institute of Science and Technology College of Engineering and Technology Department of Electronics and Communication Engineering
18ECO109J Embedded System Design using **Raspberry Pi** **Sixth Semester, 2022-23 (Even semester)**

Name :

Register Number
:

Day Order :

Venue :

Title of the Experiment :

Date of conduction
:

Date of Submission
:

Particulars	**Max. Marks**	**Marks Obtaine d**
Pre-lab / Algorithm	10	
Lab Performance	20	
Post-lab	10	
Total	**40**	

REPORT VERIFICATION

Date

:

Faculty Name :

Signature :

LAB – 7 Programming on Interrupts

AIM:

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

TASK:

1. Configure two I/O pin for LOW to HIGH transition Interrupt source and HIGH to LOW transition Interrupt source
2. 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.
3. 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 inbuilt function GPIO.add_event_detect() to detect interrupt.

ALGORITHM:

TASK 1:

1. Choose two I/O pins to be configured as Interrupt sources.
2. Set the mode of the pins to either BOARD or BCM mode using **GPIO.setmode()**.
3. Set the direction of the pins to input using **GPIO.setup()**.
4. Enable the Interrupt source for each pin using **GPIO.add_event_detect()** function.
5. Define the respective Interrupt Service Routines (ISRs) for each Interrupt source using **GPIO.add_event_callback()** function.

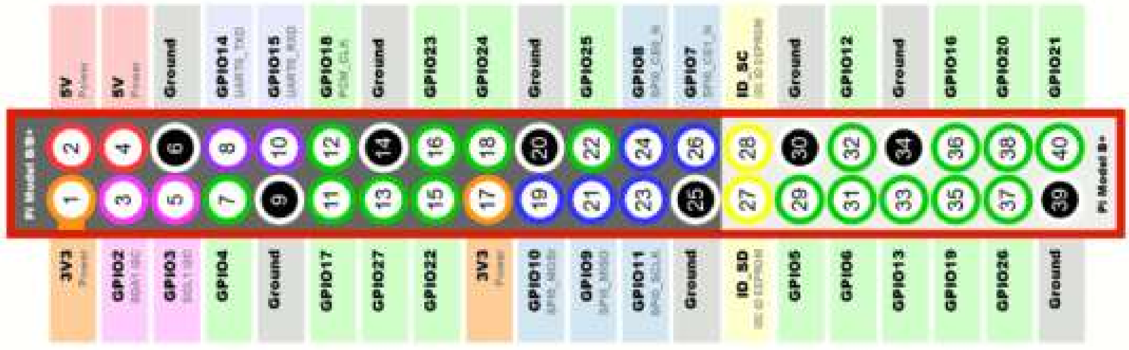
TASK 2:

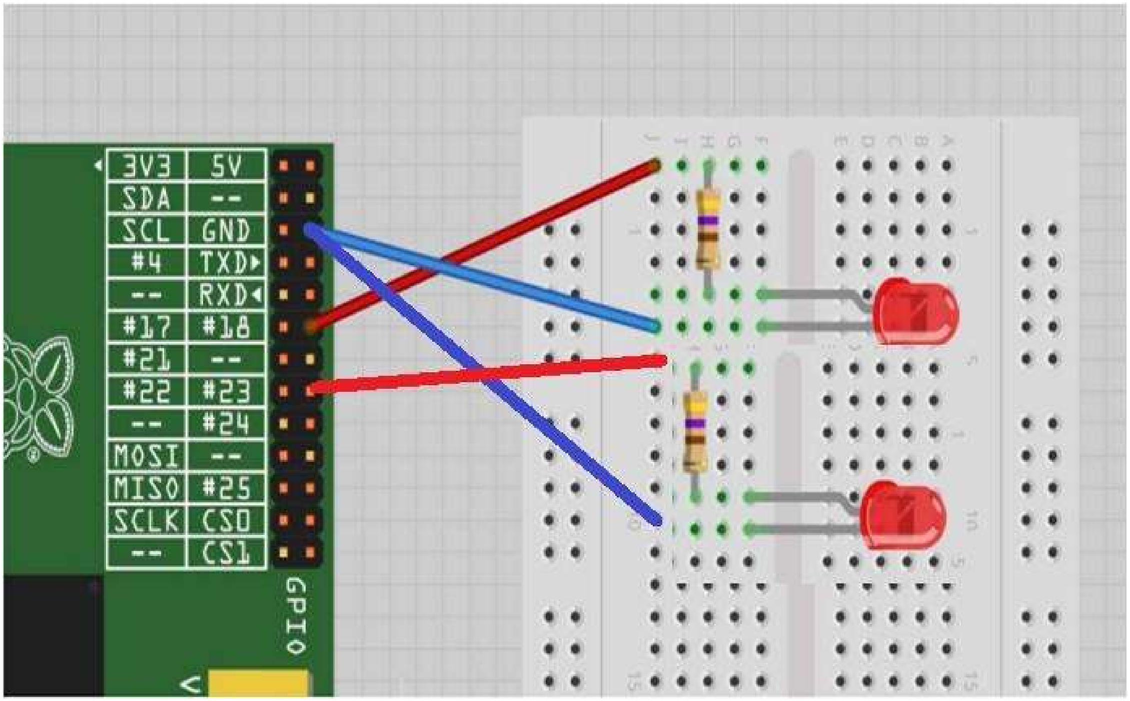
1. Define two ISR functions, ISRL2H and ISRH2L, that will be called when a LOW to HIGH transition or HIGH to LOW transition interrupt occurs, respectively.
2. In the ISRL2H function, increment a counter for positive edge interrupts and print the number of positive edges that have occurred.
3. In the ISRH2L function, increment a counter for negative edge interrupts and print the number of negative edges that have occurred.
4. Configure the I/O pins for the desired interrupt types using the appropriate GPIO function.
5. Write the main program loop that waits for an interrupt to occur.
6. When an interrupt occurs, call the appropriate ISR function to handle the interrupt and continue looping until the program is terminated.

TASK 3:

1. Define two functions for Interrupt Service Routine (ISR) for LOW to HIGH transition and HIGH to LOW transition, namely ISRL2H and ISRH2L, respectively.
2. Inside the function ISRL2H, first make ON LED1 for a while.
3. Then, print a message to indicate the occurrence of a positive edge interrupt.
4. Similarly, inside the function ISRH2L, make ON LED2 for a while.
5. Then, print a message to indicate the occurrence of a negative edge interrupt.
6. In the main program, configure two I/O pins for LOW to HIGH transition and HIGH to LOW transition interrupt source. When an interrupt signal is detected, call the appropriate ISR function based on the type of interrupt, and the corresponding LED will be turned on for a while.

PIN & CIRCUIT DIAGRAM:

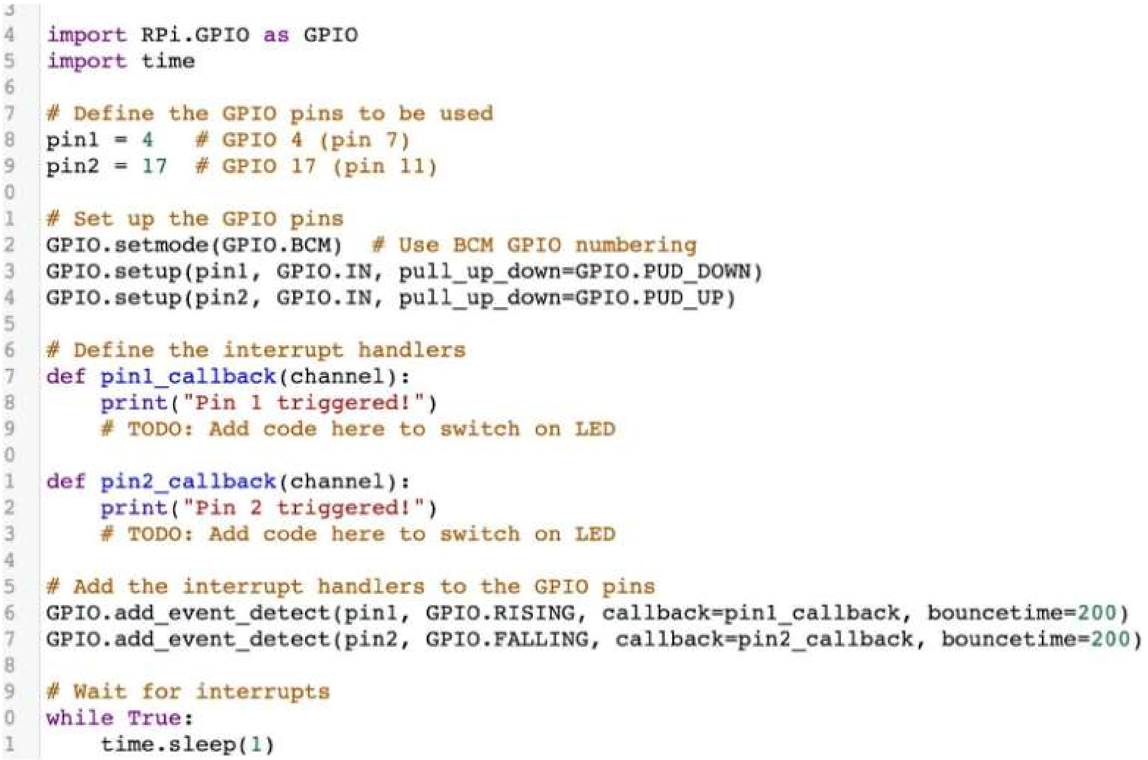




Note: Interrupt sources are not shown in the figure.

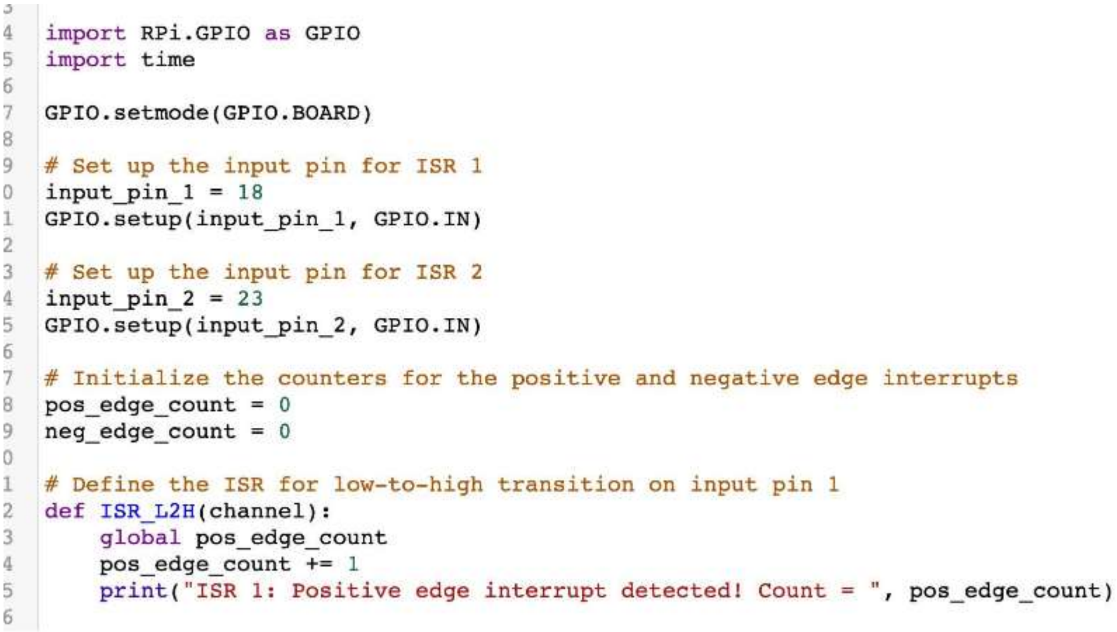
PROGRAMS
:

TASK 1:

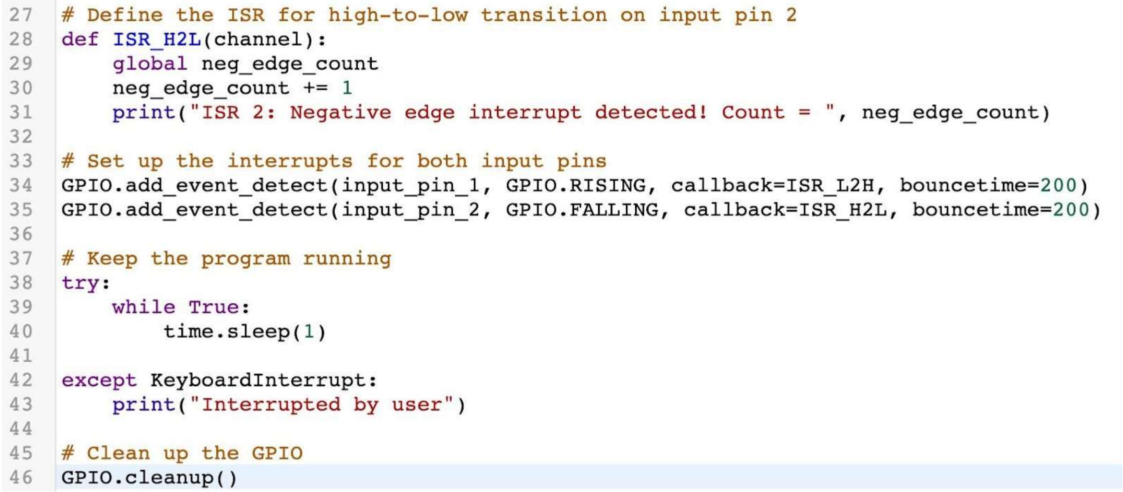


```python
import RPi.GPIO as GPIO
import time

# Define the GPIO pins to be used
pin1 = 4    # GPIO 4 (pin 7)
pin2 = 17   # GPIO 17 (pin 11)

# Set up the GPIO pins
GPIO.setmode(GPIO.BCM)  # Use BCM GPIO numbering
GPIO.setup(pin1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(pin2, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Define the interrupt handlers
def pin1_callback(channel):
    print("Pin 1 triggered!")
    # TODO: Add code here to switch on LED

def pin2_callback(channel):
    print("Pin 2 triggered!")
    # TODO: Add code here to switch on LED

# Add the interrupt handlers to the GPIO pins
GPIO.add_event_detect(pin1, GPIO.RISING, callback=pin1_callback, bouncetime=200)
GPIO.add_event_detect(pin2, GPIO.FALLING, callback=pin2_callback, bouncetime=200)

# Wait for interrupts
while True:
    time.sleep(1)
```

TASK 2:

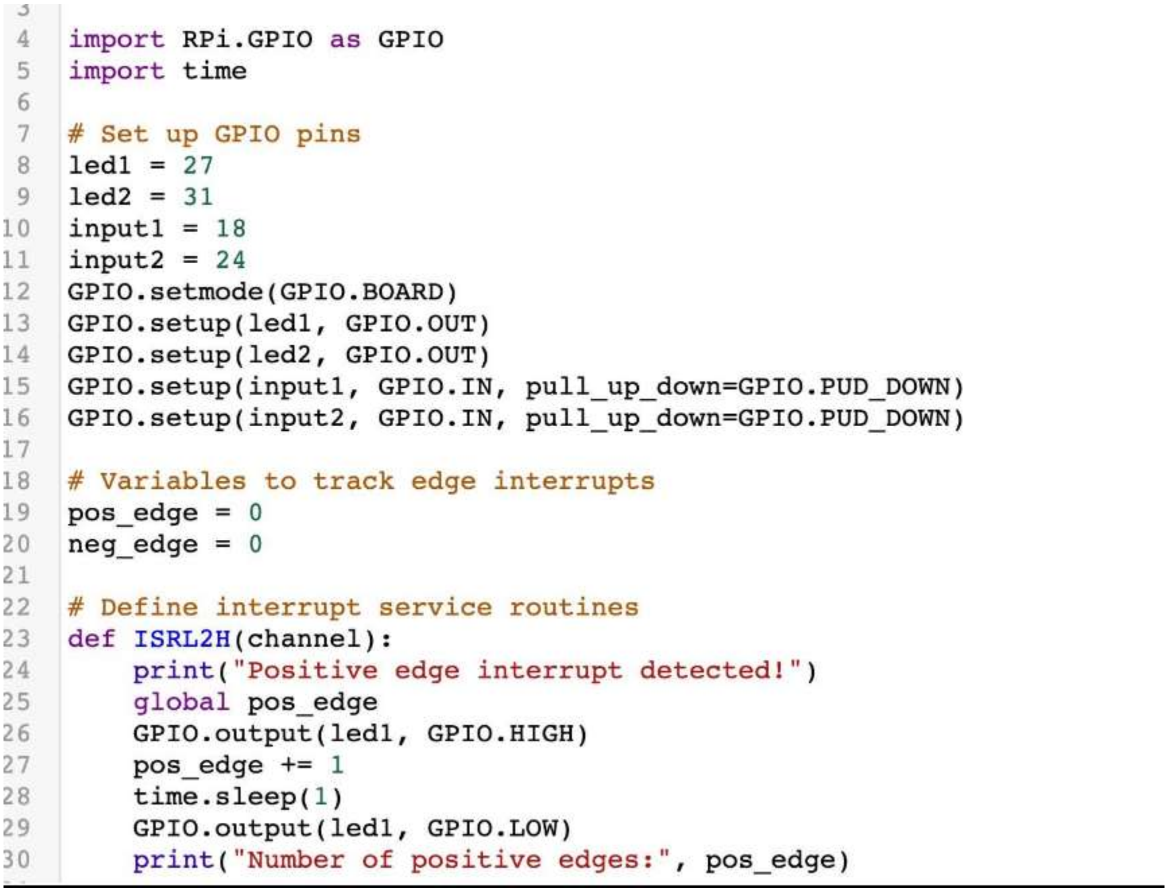


```python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

# Set up the input pin for ISR 1
input_pin_1 = 18
GPIO.setup(input_pin_1, GPIO.IN)

# Set up the input pin for ISR 2
input_pin_2 = 23
GPIO.setup(input_pin_2, GPIO.IN)

# Initialize the counters for the positive and negative edge interrupts
pos_edge_count = 0
neg_edge_count = 0

# Define the ISR for low-to-high transition on input pin 1
def ISR_L2H(channel):
    global pos_edge_count
    pos_edge_count += 1
    print("ISR 1: Positive edge interrupt detected! Count = ", pos_edge_count)

```

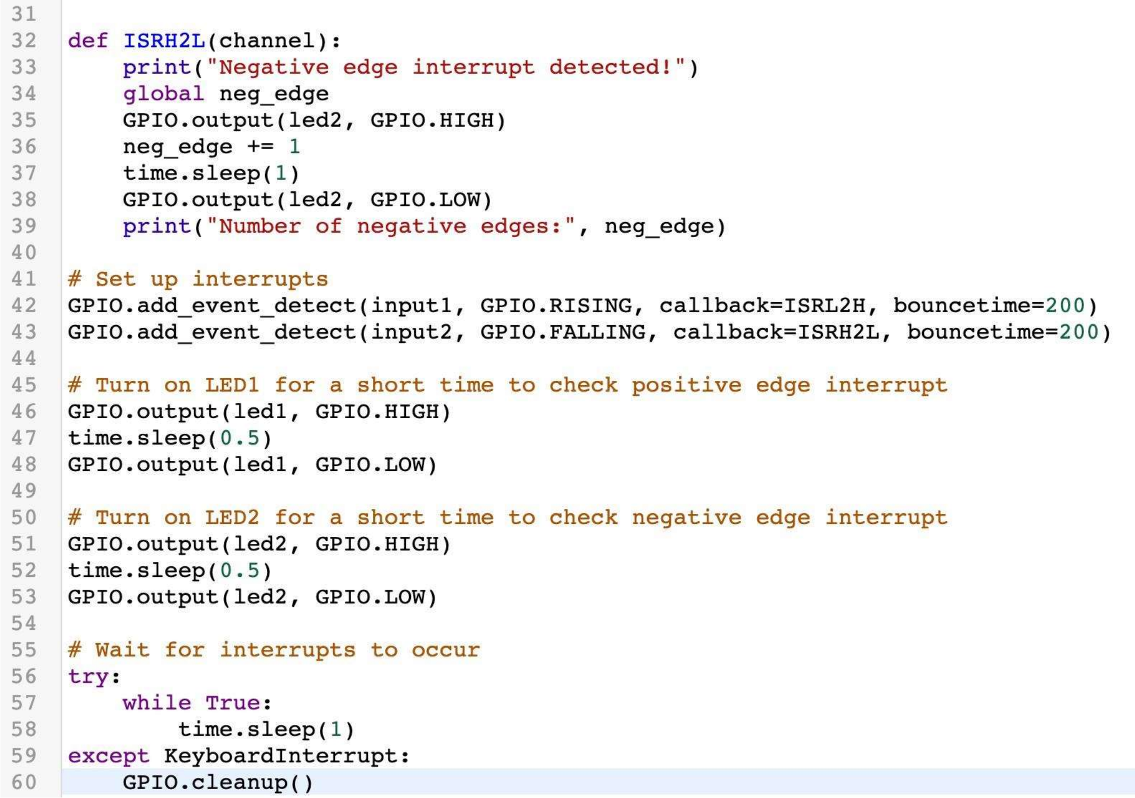


```
# Define the ISR for high-to-low transition on input pin 2
def ISR_H2L(channel):
    global neg_edge_count
    neg_edge_count += 1
    print("ISR 2: Negative edge interrupt detected! Count = ", neg_edge_count)

# Set up the interrupts for both input pins
GPIO.add_event_detect(input_pin_1, GPIO.RISING, callback=ISR_L2H, bouncetime=200)
GPIO.add_event_detect(input_pin_2, GPIO.FALLING, callback=ISR_H2L, bouncetime=200)

# Keep the program running
try:
    while True:
        time.sleep(1)

except KeyboardInterrupt:
    print("Interrupted by user")

# Clean up the GPIO
GPIO.cleanup()
```

TASK 3:



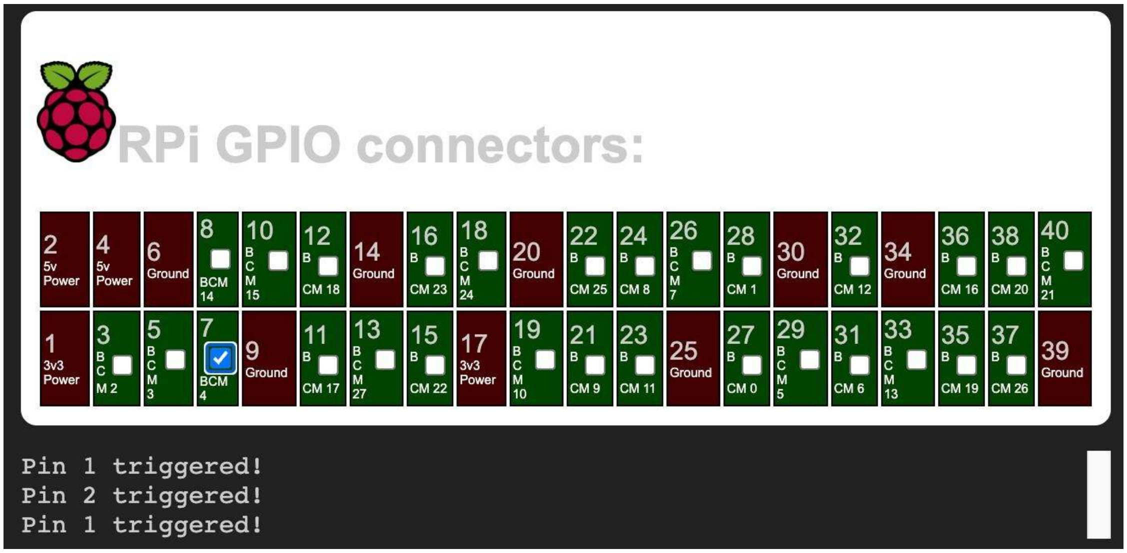
```
import RPi.GPIO as GPIO
import time

# Set up GPIO pins
led1 = 27
led2 = 31
input1 = 18
input2 = 24
GPIO.setmode(GPIO.BOARD)
GPIO.setup(led1, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)
GPIO.setup(input1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(input2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# Variables to track edge interrupts
pos_edge = 0
neg_edge = 0

# Define interrupt service routines
def ISRL2H(channel):
    print("Positive edge interrupt detected!")
    global pos_edge
    GPIO.output(led1, GPIO.HIGH)
    pos_edge += 1
    time.sleep(1)
    GPIO.output(led1, GPIO.LOW)
    print("Number of positive edges:", pos_edge)
```



```python

def ISRH2L(channel):
    print("Negative edge interrupt detected!")
    global neg_edge
    GPIO.output(led2, GPIO.HIGH)
    neg_edge += 1
    time.sleep(1)
    GPIO.output(led2, GPIO.LOW)
    print("Number of negative edges:", neg_edge)

# Set up interrupts
GPIO.add_event_detect(input1, GPIO.RISING, callback=ISRL2H, bouncetime=200)
GPIO.add_event_detect(input2, GPIO.FALLING, callback=ISRH2L, bouncetime=200)

# Turn on LED1 for a short time to check positive edge interrupt
GPIO.output(led1, GPIO.HIGH)
time.sleep(0.5)
GPIO.output(led1, GPIO.LOW)

# Turn on LED2 for a short time to check negative edge interrupt
GPIO.output(led2, GPIO.HIGH)
time.sleep(0.5)
GPIO.output(led2, GPIO.LOW)

# Wait for interrupts to occur
try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    GPIO.cleanup()
```

Output:

TASK 1:

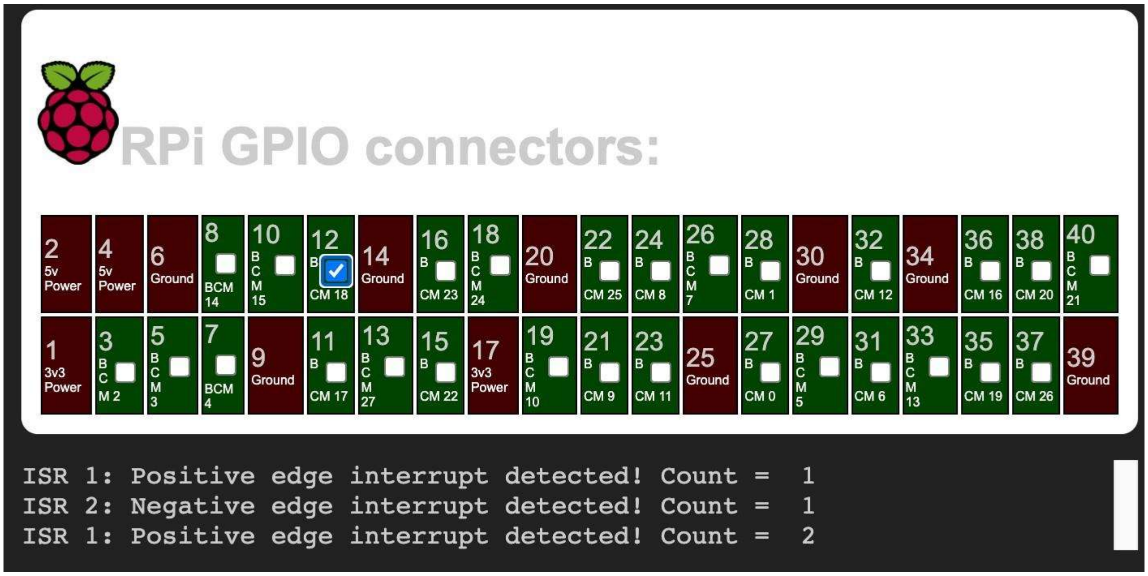


RPi GPIO connectors:

2 5v Power	4 5v Power	6 Ground	8 BCM 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 BCM 3 □	7 BCM 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

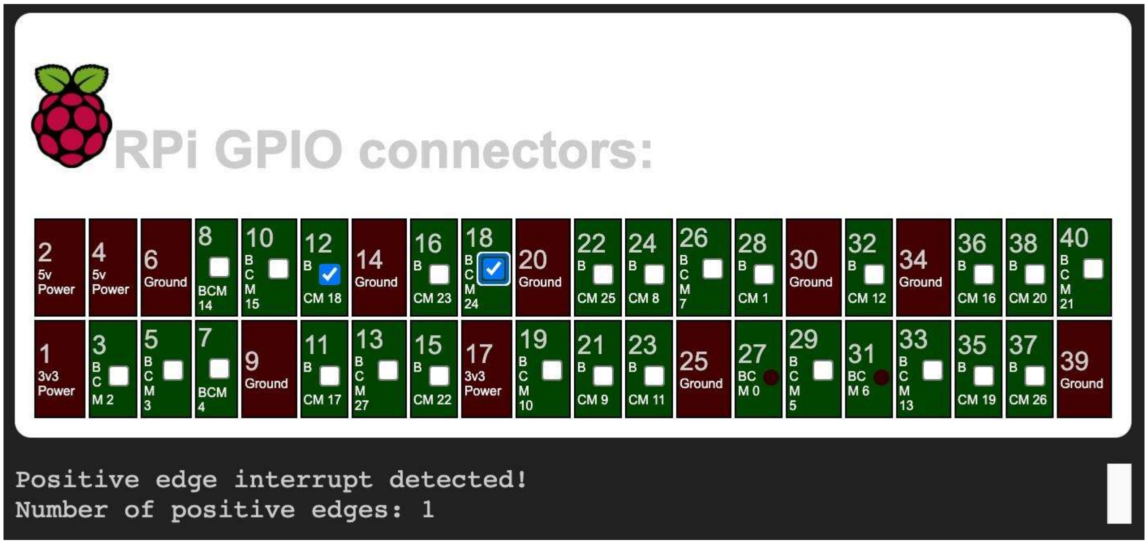
```
Pin 1 triggered!
Pin 2 triggered!
Pin 1 triggered!
```

TASK 2:



```
ISR 1: Positive edge interrupt detected! Count =  1
ISR 2: Negative edge interrupt detected! Count =  1
ISR 1: Positive edge interrupt detected! Count =  2
```

TASK 3:



```
Positive edge interrupt detected!
Number of positive edges: 1
```

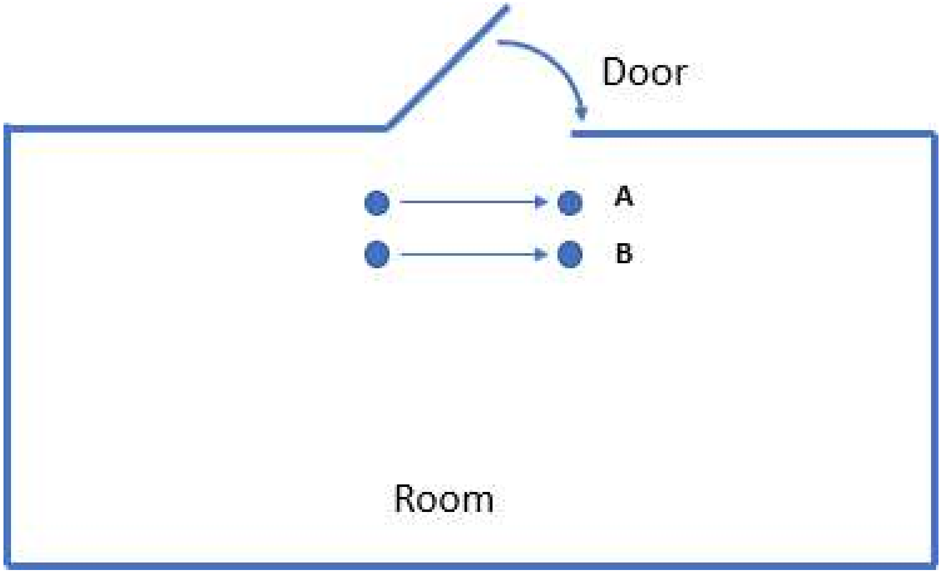
Result:

Pre-Lab Questions:

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

Post Lab Questions:

1. In a room number of persons entering and leaving has to be recorded automatically. A system is installed for this. The hardware contains two Infra-Red (IR) source and 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 below figure.



Write a suitable pseudo code or algorithm for 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.