

## 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 : Harsh Yadav

Register Number :

Day Order :

Venue :

Title of the Experiment :

Date of conduction

:

Date of Submission

:

Particulars	Max. Marks	Marks Obtained
Pre-lab / Algorithm	10	
Lab Performance	20	
Post-lab	10	
<b>Total</b>	<b>40</b>	

### 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.

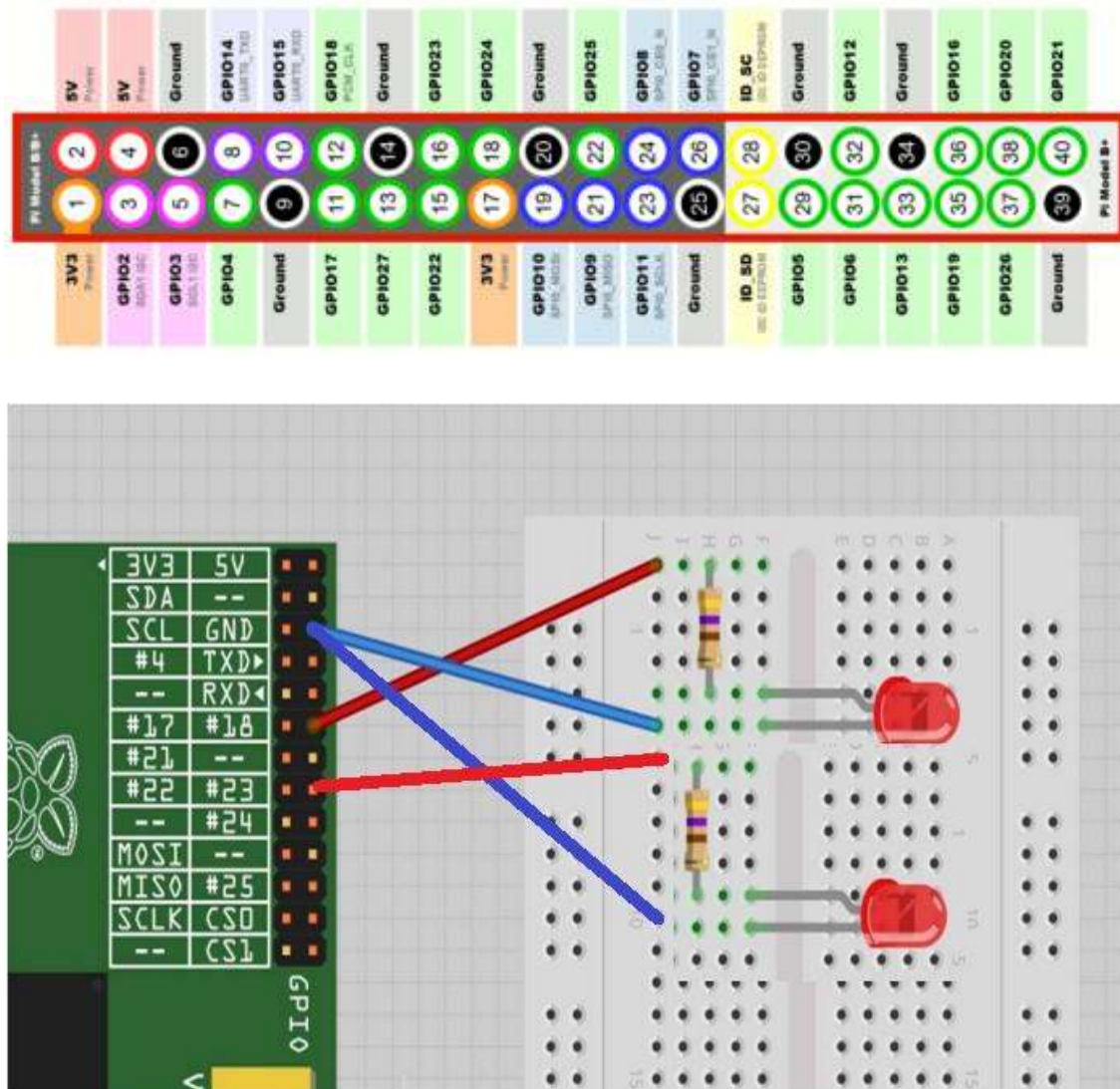
#### TASK 2:

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:

```
3
4 import RPi.GPIO as GPIO
5 import time
6
7 # Define the GPIO pins to be used
8 pin1 = 4 # GPIO 4 (pin 7)
9 pin2 = 17 # GPIO 17 (pin 11)
10
11 # Set up the GPIO pins
12 GPIO.setmode(GPIO.BCM) # Use BCM GPIO numbering
13 GPIO.setup(pin1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
14 GPIO.setup(pin2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
15
16 # Define the interrupt handlers
17 def pin1_callback(channel):
18     print("Pin 1 triggered!")
19     # TODO: Add code here to switch on LED
20
21 def pin2_callback(channel):
22     print("Pin 2 triggered!")
23     # TODO: Add code here to switch on LED
24
25 # Add the interrupt handlers to the GPIO pins
26 GPIO.add_event_detect(pin1, GPIO.RISING, callback=pin1_callback, bouncetime=200)
27 GPIO.add_event_detect(pin2, GPIO.FALLING, callback=pin2_callback, bouncetime=200)
28
29 # Wait for interrupts
30 while True:
31     time.sleep(1)
```

### TASK 2:

```
3
4 import RPi.GPIO as GPIO
5 import time
6
7 GPIO.setmode(GPIO.BOARD)
8
9 # Set up the input pin for ISR 1
10 input_pin_1 = 18
11 GPIO.setup(input_pin_1, GPIO.IN)
12
13 # Set up the input pin for ISR 2
14 input_pin_2 = 23
15 GPIO.setup(input_pin_2, GPIO.IN)
16
17 # Initialize the counters for the positive and negative edge interrupts
18 pos_edge_count = 0
19 neg_edge_count = 0
20
21 # Define the ISR for low-to-high transition on input pin 1
22 def ISR_L2H(channel):
23     global pos_edge_count
24     pos_edge_count += 1
25     print("ISR 1: Positive edge interrupt detected! Count = ", pos_edge_count)
26
```

```

27 # Define the ISR for high-to-low transition on input pin 2
28 def ISR_H2L(channel):
29     global neg_edge_count
30     neg_edge_count += 1
31     print("ISR 2: Negative edge interrupt detected! Count = ", neg_edge_count)
32
33 # Set up the interrupts for both input pins
34 GPIO.add_event_detect(input_pin_1, GPIO.RISING, callback=ISR_L2H, bouncetime=200)
35 GPIO.add_event_detect(input_pin_2, GPIO.FALLING, callback=ISR_H2L, bouncetime=200)
36
37 # Keep the program running
38 try:
39     while True:
40         time.sleep(1)
41
42 except KeyboardInterrupt:
43     print("Interrupted by user")
44
45 # Clean up the GPIO
46 GPIO.cleanup()

```

### TASK 3:

```

4 import RPi.GPIO as GPIO
5 import time
6
7 # Set up GPIO pins
8 led1 = 27
9 led2 = 31
10 input1 = 18
11 input2 = 24
12 GPIO.setmode(GPIO.BOARD)
13 GPIO.setup(led1, GPIO.OUT)
14 GPIO.setup(led2, GPIO.OUT)
15 GPIO.setup(input1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
16 GPIO.setup(input2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
17
18 # Variables to track edge interrupts
19 pos_edge = 0
20 neg_edge = 0
21
22 # Define interrupt service routines
23 def ISRL2H(channel):
24     print("Positive edge interrupt detected!")
25     global pos_edge
26     GPIO.output(led1, GPIO.HIGH)
27     pos_edge += 1
28     time.sleep(1)
29     GPIO.output(led1, GPIO.LOW)
30     print("Number of positive edges:", pos_edge)

```



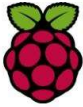
```

31
32 def ISRH2L(channel):
33     print("Negative edge interrupt detected!")
34     global neg_edge
35     GPIO.output(led2, GPIO.HIGH)
36     neg_edge += 1
37     time.sleep(1)
38     GPIO.output(led2, GPIO.LOW)
39     print("Number of negative edges:", neg_edge)
40
41 # Set up interrupts
42 GPIO.add_event_detect(input1, GPIO.RISING, callback=ISRL2H, bouncetime=200)
43 GPIO.add_event_detect(input2, GPIO.FALLING, callback=ISRH2L, bouncetime=200)
44
45 # Turn on LED1 for a short time to check positive edge interrupt
46 GPIO.output(led1, GPIO.HIGH)
47 time.sleep(0.5)
48 GPIO.output(led1, GPIO.LOW)
49
50 # Turn on LED2 for a short time to check negative edge interrupt
51 GPIO.output(led2, GPIO.HIGH)
52 time.sleep(0.5)
53 GPIO.output(led2, GPIO.LOW)
54
55 # Wait for interrupts to occur
56 try:
57     while True:
58         time.sleep(1)
59 except KeyboardInterrupt:
60     GPIO.cleanup()

```

Output:

TASK 1:



## RPi GPIO connectors:


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

Pin 1 triggered!

Pin 2 triggered!

Pin 1 triggered!

TASK 2:

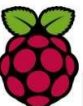


### RPi GPIO connectors:

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

```
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:



### 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

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

## 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	
<b>Total</b>	<b>40</b>	

### REPORT VERIFICATION

Date :

Faculty Name :

Signature :



## LAB – 8 Programming on Stepper Motor

### AIM:

To write a program to run a stepper motor in Python.

### TASK:

1. Write a Python program to run a stepper motor in Python with following conditions.  
Assume unipolar stepper motor. The step sequence should be '1010', '0110', '0101', '1001'. The term '1' represents excitation of stepper motor winding. The motor has to move 'forward' and 'reverse' based on user request. Assume number of steps as 2. Print the winding sequence as well as direction i.e. forward or reverse.

### ALGORITHM:

1. Set four pins in 'Board' mode as output pins
2. Create forward and reverse sequence.
3. Get the direction and number of steps.
4. If the direction is forward then use forward sequence to excite the winding by sending '1' in appropriate pins. If the direction is reverse then use reverse sequence to excite the winding.

### PIN & CIRCUIT DIAGRAM:



Figure 1. Pin Configuration of Board

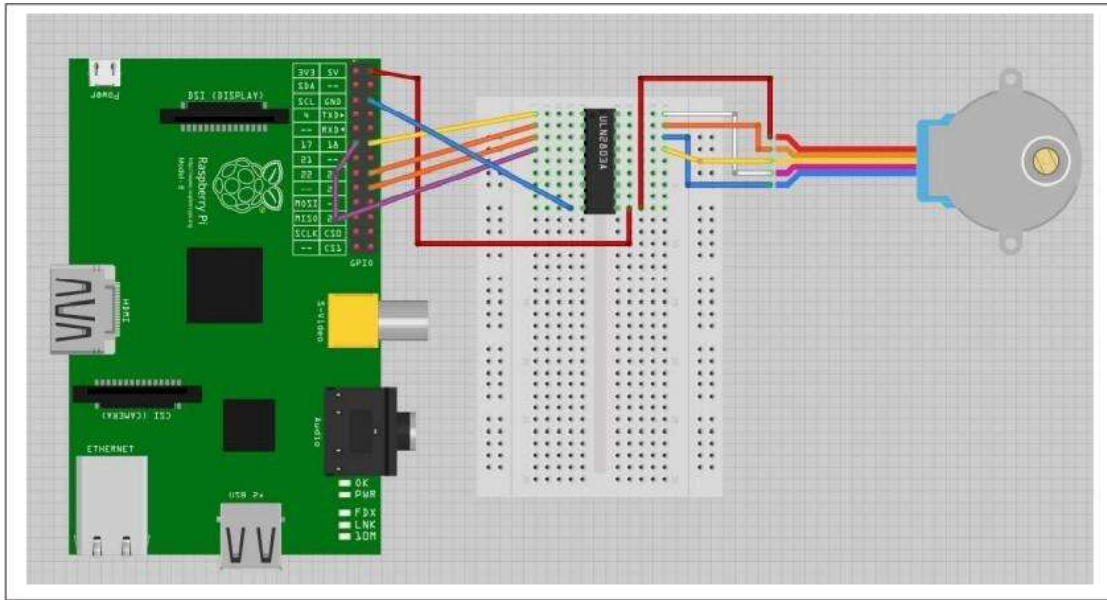


Figure 2. Raspberry Pi interfaced with unipolar stepper motor

## PROGRAMS:

```
3 import RPi.GPIO as GPIO
4 import time
5
6 # Define the GPIO pins for the stepper motor
7 IN1 = 17
8 IN2 = 18
9 IN3 = 27
10 IN4 = 22
11
12 # Define the step sequence
13 SEQ = ['1010', '0110', '0101', '1001']
14
15 # Set up the GPIO pins
16 GPIO.setmode(GPIO.BCM)
17 GPIO.setwarnings(False)
18 GPIO.setup(IN1, GPIO.OUT)
19 GPIO.setup(IN2, GPIO.OUT)
20 GPIO.setup(IN3, GPIO.OUT)
21 GPIO.setup(IN4, GPIO.OUT)
22
23 # Define a function to set the motor winding sequence
24 def set_sequence(seq):
25     GPIO.output(IN1, int(seq[0]))
26     GPIO.output(IN2, int(seq[1]))
27     GPIO.output(IN3, int(seq[2]))
28     GPIO.output(IN4, int(seq[3]))
29
30 # Define a function to move the motor forward
31 def move_forward(steps):
32     print("Moving forward...")
33     for i in range(steps):
34         for seq in SEQ:
35             set_sequence(seq)
36             time.sleep(0.01)
37
```

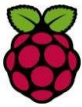
```

37
38 # Define a function to move the motor in reverse
39 def move_reverse(steps):
40     print("Moving in reverse...")
41     for i in range(steps):
42         for seq in reversed(SEQ):
43             set_sequence(seq)
44             time.sleep(0.01)
45
46 # Prompt the user for the direction and number of steps
47 direction = input("Enter the direction (forward/reverse): ")
48 steps = int(input("Enter the number of steps: "))
49
50 # Move the motor in the specified direction
51 if direction == "forward":
52     move_forward(steps)
53 elif direction == "reverse":
54     move_reverse(steps)
55 else:
56     print("Invalid direction")
57
58 # Clean up the GPIO pins
59 GPIO.cleanup()

```

OUTPUT:

mycode.py



## 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

Enter the direction (forward/reverse): forward

Enter the number of steps: 2

Moving forward...

>\_ REPL



## 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	
<b>Total</b>	<b>40</b>	

### REPORT VERIFICATION

Date :

Faculty Name :

Signature :



## LAB – 9 Programming on Switch

### AIM:

To write a program to utilize a push switch to control an LED.

### TASK:

1. To turn an LED on and off with a push switch so that it toggles between on and off each time switch is pressed.

### ALGORITHM:

1. Set one pin as input and another pin as output in 'Board' mode.
2. Initialize the state of LED as False and state of input state as True.
3. Read the input pin and store the Boolean value in a variable (say new\_input\_state)
4. Check whether new\_input\_state is False and the old input state is True. If yes then change the state of the LED.
5. Wait for a while.
6. Repeat the steps 4 and 5 continuously.

### Pin & Circuit Diagram:



Figure 1. Pin Configuration of Board

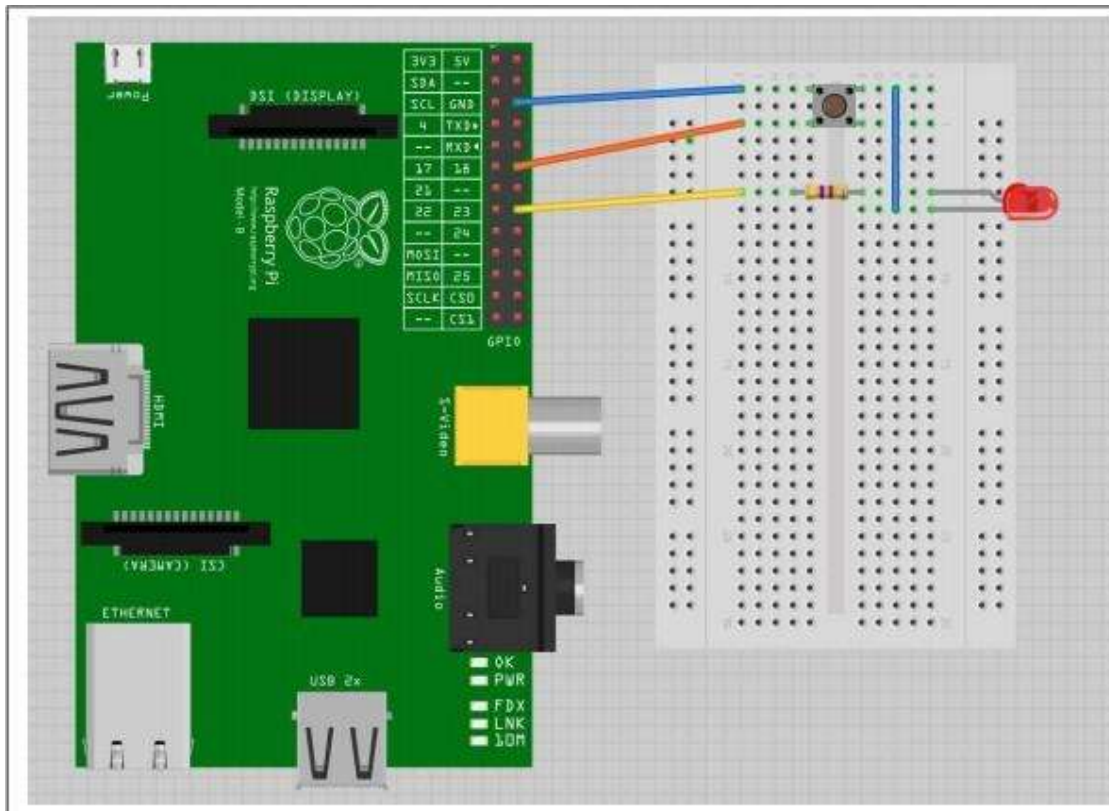


Figure 2. Raspberry Pi interfaced with a Switch and LED

## PROGRAM:

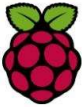
```

3 import RPi.GPIO as GPIO
4 import time
5
6 # Define the GPIO pins for the LED and switch
7 LED_PIN = 18
8 SWITCH_PIN = 17
9
10 # Set up the GPIO pins
11 GPIO.setmode(GPIO.BCM)
12 GPIO.setwarnings(False)
13 GPIO.setup(LED_PIN, GPIO.OUT)
14 GPIO.setup(SWITCH_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
15
16 # Define a function to toggle the LED
17 def toggle_led(channel):
18     if GPIO.input(LED_PIN):
19         GPIO.output(LED_PIN, GPIO.LOW)
20     else:
21         GPIO.output(LED_PIN, GPIO.HIGH)
22
23 # Set up the switch to trigger the toggle_led function on a falling edge (i.e. when the switch is pressed)
24 GPIO.add_event_detect(SWITCH_PIN, GPIO.FALLING, callback=toggle_led, bouncetime=200)
25
26 # Loop forever, waiting for events
27 while True:
28     time.sleep(1)
29
30 # Clean up the GPIO pins
31 GPIO.cleanup()

```

OUTPUT:

mycode.py



# 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

## 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	
<b>Total</b>	<b>40</b>	

### REPORT VERIFICATION

Date :

Faculty Name :

Signature :

## **LAB – 10 Programming on LED Matrix Display**

### **AIM:**

To write a python program to display a given symbol pattern and text on the LED Matrix Display on Sense HAT platform

### **TASK:**

1. To write a python program to display(scrolling) “RASPBERRY PI ” in sensehat LED matrix display and execute it in the Sense HAT simulator
2. To write a python program to display “give way” traffic symbol and diode symbol and execute it in the Sense HAT simulator

### **ALGORITHM:**

#### **TASK 1 :**

1. Import the necessary libraries: In this case, you need to import the sense\_hat library to interface with the Sense HAT LED matrix.
2. Define the scrolling message: Define a variable to hold the message you want to scroll across the LED matrix. In this case, the message is "RASPBERRY PI".
3. Set up the scrolling speed and text color: Define variables to hold the speed at which the message scrolls across the LED matrix and the color of the text. In this case, a scrolling speed of 0.05 seconds and a green text color are used.
4. Display the scrolling message: Use the show\_message() method from the sense\_hat library to display the scrolling message on the LED matrix. Pass in the message, speed, and text color as arguments to the show\_message() method.
5. Run the program in the Sense HAT simulator: Use the Sense HAT simulator to run the Python program and display the scrolling message on the virtual LED matrix.

#### **TASK 2:**

1. Import the necessary libraries: In this case, you need to import the sense\_hat library to interface with the Sense HAT LED matrix.
2. Define the symbols: Define two variables to hold the symbols you want to display on the LED matrix. In this case, the "give way" symbol can be represented by an arrow pointing to the right, and the diode symbol can be represented by a filled-in square.
3. Set up the LED matrix display: Use the set\_pixels() method from the sense\_hat library to display the symbols on the LED matrix. Create a 2D list of colors representing the LED matrix display, where each element of the list represents a single LED on the matrix.
4. Display the symbols on the LED matrix: Pass the 2D list of colors to the set\_pixels() method to display the symbols on the LED matrix.
5. Run the program in the Sense HAT simulator: Use the Sense HAT simulator to run the Python program and display the symbols on the virtual LED matrix.

PIN & CIRCUIT DIAGRAM:



Figure 1. Pin Configuration of Board

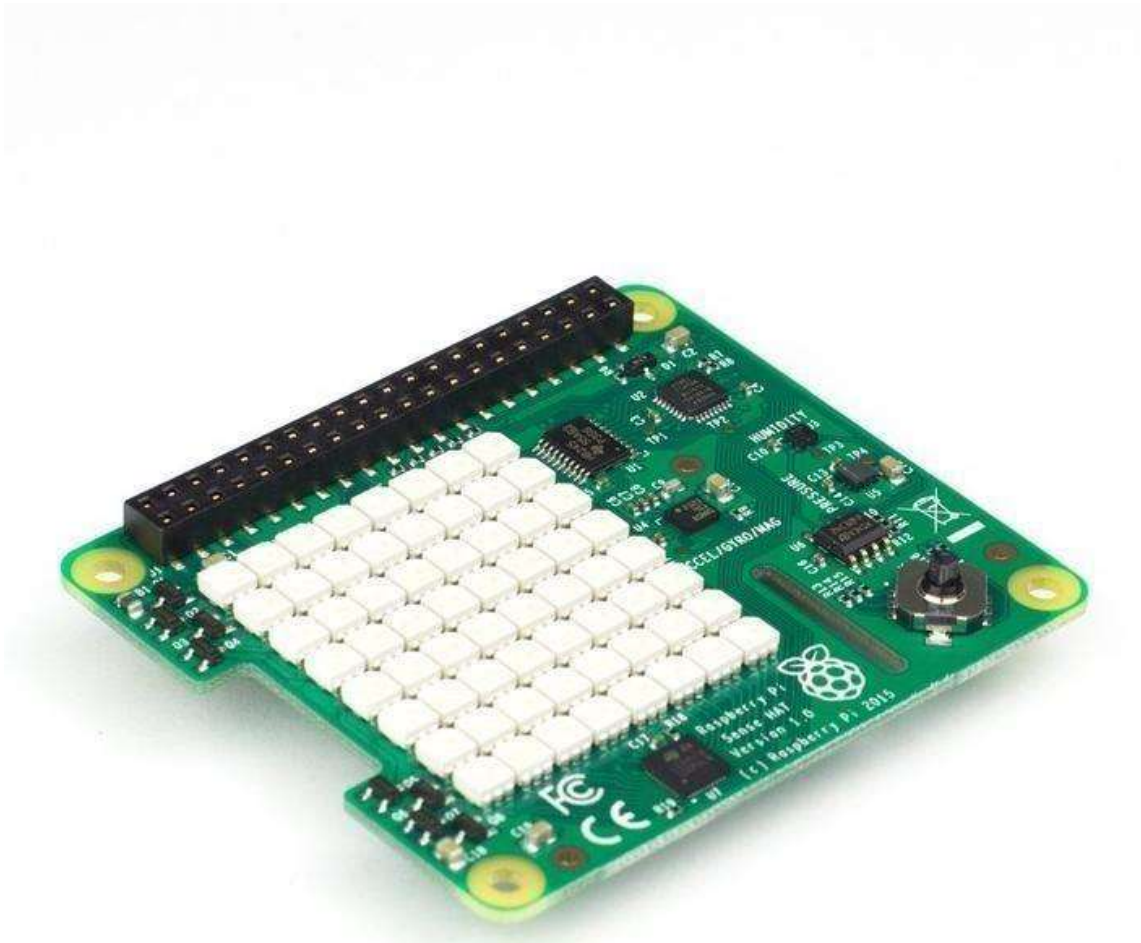


Figure 2. Raspberry Pi SenseHat board with 8x8 matrix LED display



## PROGRAM

:

### TASK 1:

```
from sense_hat import SenseHat
import time

sense = SenseHat()

# Set up colors
red = (255, 0, 0)
green = (0, 255, 0)

# Set up message to scroll
message = "RASPBERRY PI"

# Set up scrolling speed and text color
speed = 1
text_color = green

# Scroll the message continuously
while True:
    sense.show_message(message, speed, text_color)
```

### TASK 2:

```
3 from sense_hat import SenseHat
4 import time
5
6 sense = SenseHat()
7
8 # Set up colors
9 red = (255, 0, 0)
10 yellow = (255, 255, 0)
11 black = (0, 0, 0)
12 white = (255, 255, 255)
13
14 # Define images
15 give_way = [
16     black, black, black, red, red, black, black, black,
17     black, black, red, yellow, yellow, red, black, black,
18     black, red, yellow, white, white, yellow, red, black,
19     red, yellow, white, white, white, white, yellow, red,
20     red, yellow, white, white, white, white, yellow, red,
21     black, red, yellow, white, white, yellow, red, black,
22     black, black, red, yellow, yellow, red, black, black,
23     black, black, black, red, red, black, black, black
24 ]
```

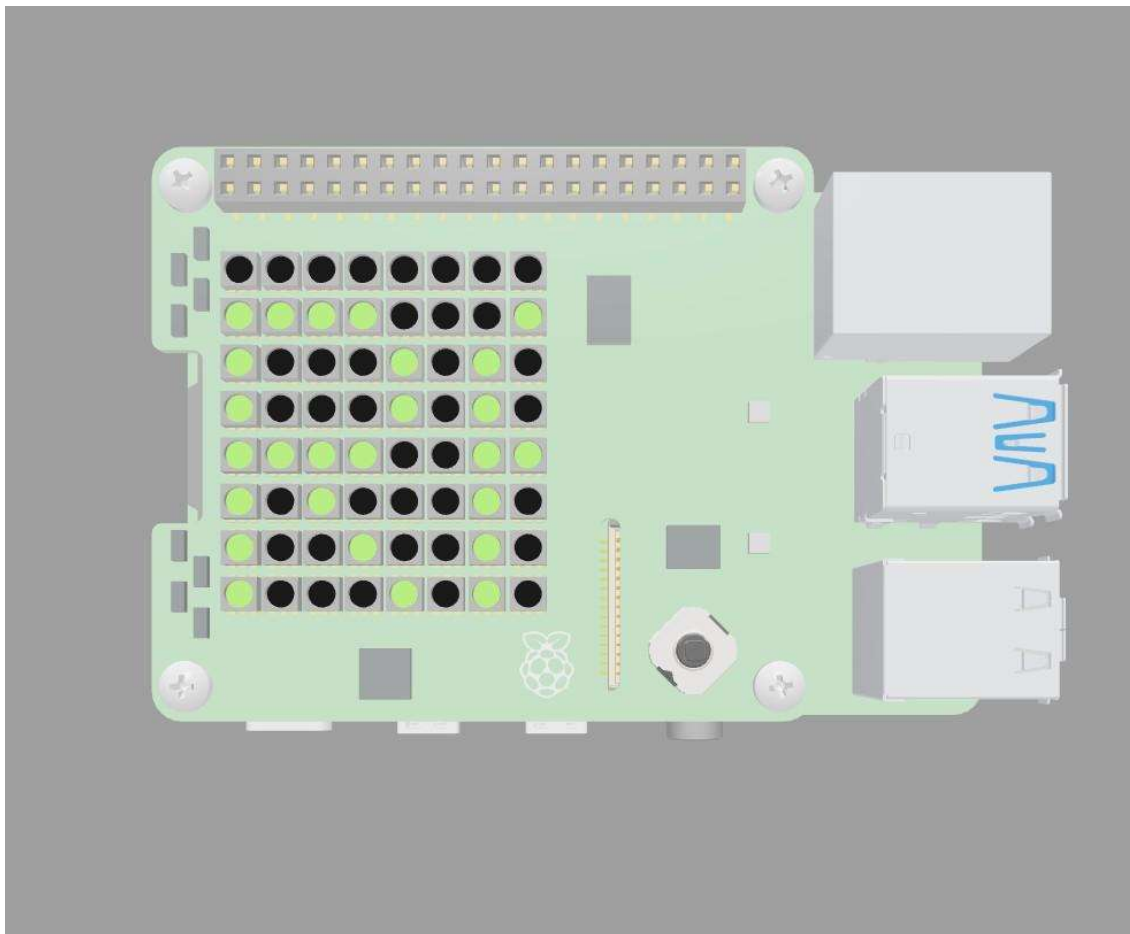
```

26 diode = [
27     black, black, black, white, white, black, black, black,
28     black, black, black, white, white, black, black, black,
29     black, black, black, white, white, black, black, black,
30     black, black, black, white, white, black, black, black,
31     black, black, black, white, white, black, black, black,
32     black, black, black, white, white, black, black, black,
33     black, black, black, white, white, black, black, black,
34     black, black, black, white, white, black, black, black
35 ]
36
37 # Display give_way symbol
38 sense.set_pixels(give_way)
39 time.sleep(5)

```

**OUTPUT:**

**TASK 1:**



**TASK 2:**

