EXPERIMENT NUMBER : 1

EXPERIMENT NAME : GENERAL PURPOSE INPUT & OUTPUT PROGRAMMING USING RASPBERRYPI WITH SENSEHAT

DATE : 26/09/2022, MONDAY

8/10

* **AIM :**

To introduce Python-based basic programs using Raspberry Pi and GPIO and associated peripheral interfacing.

* **INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) :**

Name - Thonny 4.0.1

Publisher - Aivar Annamaa

Support link - https://thonny.org

* **IMPORT NECESSARY LIBRARIES :**

```
from sense_hat import SenseHat # Python module to control the Raspberry Pi Sense HAT
sense = SenseHat()
sense.clear() # No arguments defaults to OFF

from time import sleep # Handle time-related tasks
```

(a) Hello World Display on SenseHAT:

→ Algorithm-

```
① Initialize the required colours for text and background.
② Controll the scroll speed; Greater speed implies slower scroll time.
③ Wait for some seconds.
④ Clear everything.
```

→ Python Code -

```
yellow = (255, 255, 0)
blue = (0, 0, 255)
```

(b) single character Displayed followed by Rotation and Flip:

→ Algorithms -

① Display a single text character on the LED matrix.
② Choose a suitable text and background colour
③ Perform horizontal and vertical flip.
④ Rotate the character by some degree of your choice.

→ Python Code-

```
sense.show_letter ("Z",
                   text_colour = yellow, # Text Colour
                   back_colour = blue) # Background Colour

sense.flip_h() # Horizontal Flip
sense.flip_v() # Vertical Flip
sense.set_rotation (180) # Rotate character by 180 degrees
```

(c) Single Pixel activation in SenseHat:

→ Algorithm -

① Choose Column Number.
② Choose Row Number.
③ Choose suitable colour.

→ Python Code -

```
sense.set_pixel (1, 2, blue) # First Column, Second Row, Blue Colour
```

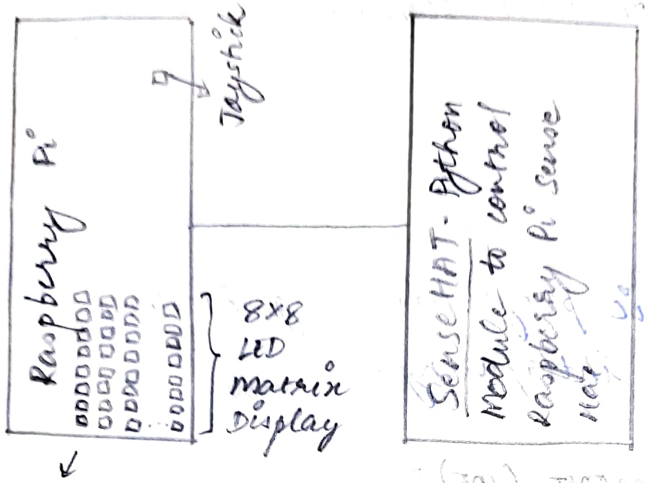
(d) custom character Creation using LED Matrix: (Print the letter 'S')

→ Algorithm -

① Choose Column Number.
② Choose Row Number.
③ Choose suitable colour.

→ Python Code -

```
sense.set_pixel (0,0, blue); sense.set_pixel (1,0, blue); sense.set_pixel
(2,0, blue); sense.set_pixel (3,0, blue); sense.set_pixel (4,0, blue); sense.set_pixel (5,0, blue);
sense.set_pixel (6,0, blue); sense.set_pixel (7,0, blue)
```



contains other sensors as well such as gyroscope, humidity, temperature, pressure, magnetometer, accelerometer, etc.

```
sense. set_pixel (0,1, blue); sense. set_pixel (0,2, blue)
sense. set_pixel (0, 3, blue); sense. set_pixel (1,3, blue); sense. set_pixel
(2,3, blue); sense. set_pixel (3,3, blue); sense. set_pixel (4,3, blue);
sense. set_pixel (5,3, blue); sense. set_pixel (6,3, blue); sense. set_pixel
(7, 3, blue)
sense. set_pixel (0,4, blue); sense. set_pixel (1,4, blue); sense. set_pixel
(2,4, blue); sense. set_pixel (3,4, blue); sense. set_pixel (4, 4, blue);
sense. set_pixel (5,4, blue); sense. set_pixel (6,4, blue); sense. set_pixel
(7, 4, blue)
sense. set_pixel (7,5, blue); sense. set_pixel (7,6, blue)
sense. set_pixel (0,7, blue); sense. set_pixel (1,7, blue); sense. set_pixel
(2,7, blue); sense. set_pixel (3,7, blue); sense. set_pixel (4,7, blue);
sense. set_pixel (5,7, blue); sense. set_pixel (6,7, blue); sense. set_pixel
(7,7, blue)
```

(e) **Position and Action Display of Joystick:**

→ Algorithm -

① Return a list of InputEvent tuples representing all events that have occurred since the last call to get_events or wait_for_event.

② Direction - The direction the joystick was moved, as a string. ("up", "down", "left", "right", "middle")

③ Action - The action that occurred, as a string ("pressed", "released", "held")

④ Print the event direction and action.

→ Python Code -

```
while True:
    for event in sense.stick.get_events():
        print (event. direction, event. action)
```

(f) **Character Display based on Position and Action of Joystick:**

→ Algorithm -

① Return a list of InputEvent tuples representing all events that have occurred since the last call to get_events or wait_for_event.

② If the direction in which the joystick was moved is "middle" and the action that occurred is "held", then -
③ Display a single text character on the LED matrix.
④ choose suitable text and background colour.

→ Python Code -

```
while True:
    for event in sense.stick.get_events():
        if (event.direction == "middle" and event.action == "held"):
            sense.show_letter ("z",
                              text_colour = yellow, # Text Colour
                              back_colour = blue) # Background Colour
```

(g) SenseHAT Led Matrix control based on Position and Action of Joystick using Function:

→ Algorithm -

① Define functions that sets the entire LED matrix to red, green, blue and yellow colours.
② Tell the program which function to associate with which direction.
③ Return a list of InputEvent tuples representing all events that have occurred since the last call to get_events or wait_for_events.
④ If the direction in which the joystick was moved is "middle" and the action that occurred is "held", then -
(i) Initialize the required colours.
(ii) Display a single text character on the LED matrix.
(iii) Choose suitable text and background colour.
⑤ Else, clear everything.
⑥ Keep the program running to receive the joystick events.

→ Python Code -

```
def red(): # Sets the entire LED matrix to red colour
    sense.clear (255, 0, 0)
```

```
def green (): # sets the entire LED matrix to green colour
    sense. clear (0, 255, 0)

def blue (): # sets the entire LED matrix to blue colour
    sense. clear (0, 0, 255)

def yellow (): # sets the entire LED matrix to yellow colour
    sense. clear (255, 255, 0)

# Tell the program which function to associate with which direction :-
sense. stick . direction _ up = red
sense. stick . direction _ down = blue
sense. stick . direction _ left = green
sense. stick . direction _ right = yellow
sense. stick. direction _ middle = sense. clear # Press the enter key

# Digital Input operations using Joystick :-
while True:
    for event in sense. stick. get_events ():
        print (event . direction , event. action)

    if (event . direction == "middle" and event. action == "held"):
        yellow = (255, 255, 0)~~;~~ blue = (0, 0, 255) # Initialize the
colours
            sense. show_letter ("Z",
                                text_colour = yellow, # Text colour
                                back_colour = blue) # Background Colour
    else :
        sense. clear () # Clear everything

while True:
    pass # This keeps the program running to receive the
joystick events.
```

(4) Assignment: To implement a counter which displays the count on the led matrix. The count need to be incremented on pressing the joystick's right key, it should get decremented on pressing the joystick's left key and should get cleared to zero when the joystick's middle key is pressed.

→ Algorithm-

① Initialize the counter variable to zero.
② Initialize the required colours for text and background display.
③ Define functions to increment / decrement the counter value.
(i) Declare the count variable globally.
(ii) Increment / Decrement the counter value.
(iii) Display the value with suitable text and background display.
④ Tell the program which function to associate with which direction.
⑤ Keep the program running to receive the joystick events.

→ Python Code -

```
count = 0
yellow = (255, 255, 0)
blue = (0, 0, 255)


def increment_value():
    global count
    count = count + 1
    sense.show_letter (str (count),
                        text_colour = yellow, # Text colour
                        back_colour = blue) # Background Colour


def decrement_value():
    global count
    count = count + 1
    sense.show_letter (str (count),
                        text_colour = yellow, # Text Colour
                        back_colour = blue) # Background Colour
```

```
# Tell the program which function to associate with which
direction :-
sense. stick. direction_up = increment_value
sense. stick. direction_down = decrement_value
sense. stick. direction_middle = sense. clear # Press the enter key

while True:
    pass # This keeps the program running to receive the
joystick events
```

+ **RESULT:**

Thus, introduced Python basic programs using Raspberry Pi and GPIO and associated peripheral interfacing. All the simulation results were verified successfully.

[illegible] 10/11