7/10

EXPERIMENT NUMBER : 2

EXPERIMENT NAME: SENSOR INTERFACING AND DATA LOGGING USING RASPBERRY PI WITH SENSE HAT

DATE: 08/10/2022 , SATURDAY

* **AIM:**

To perform sensor interfacing and data logging using Raspberry Pi.

* **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
```

* **INITIALIZE THE COLOURS:**

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

(a) Humidity Sensor Programming:

Gets the percentage of relative humidity from the humidity sensor.

→ Python Code -

```
humidity = sense.get_humidity()
print("Relative Humidity: " + str(round(humidity, 2)) + "%")
```

(b) Pressure sensor programming:

Gets the current pressure in millibars from the pressure sensor.

→ Python Code -

```
pressure = sense.get_pressure()
print("Current Pressure: " + str(round(pressure, 2)) + " millibars")
```

(c) Temperature Sensor Programming:

temperature in degrees from the temperature sensor.

→ Python Code -

```
temperature = sense.get_temperature()
print("Temperature: " + str(round(temperature, 2)) + " degree(s)")
```

(d) Print Local Time:

① Return the time in seconds since the epoch as a floating point number. (no section available)

② Convert a time expressed in seconds since the epoch to a string of a form 'Sun Jun 20 23:21:05 1993' representing local time.

→ Python Code -

```
seconds = time.time()
local_time = time.ctime(seconds)
print("Local Time: " + str(local_time))
```

(e) Temperature Controller using SenseHAT:-

① check if the temperature is greater than 34 (some) degrees.

② If yes, show the message in the led matrix, control scroll speed and choose suitable text and background colours.

③ Print that the temperature limit has been crossed.

④ Clear the display.

→ Python code -

```
if (temperature > 34):
    sense.show_message ("Temperature Limit Crossed!",
                        scroll_speed = 0.005,
                        text_colour = green,
                        back_colour = red)
    print ("Temperature Limit Crossed!")
sense.clear() # No arguments defaults to OFF
```

(4) Accelerometer Sensor Programming:

① Calls set_imu_config to disable the magnetometer and gyroscope.

② Then, gets the current orientation from the accelerometer only.

③ Print the values of pitch, roll and yaw and show the same on the LED matrix.

→ Python code -

```
accel_only = sense.get_accelerometer()
print ("Pitch: {pitch}, Roll: {roll}, Yaw: {yaw}".format
(**accel_only))

sense.show_message ("Pitch: {pitch}, Roll: {roll}, Yaw: {yaw}".
                    format (**accel_only),
                    scroll_speed = 0.005,
                    text_colour = yellow,
                    back_colour = blue)
```

(5) Acceleration Detection using Accelerometer:

① Gets the raw x, y and z axis accelerometer data.

② Print the x, y and z values.

③ Display the same on the LED matrix.

→ PTO

→ Python Code -

```
raw = sense.get_accelerometer_raw() # Gets the raw x, y and z
axis accelerometer data.
print ("x: {x}, y: {y}, z: {z}".format(**raw))

sense.show_message ("x: {x}, y: {y}, z: {z}".format(**raw),
                    scroll_speed = 0.005,
                    text_colour = yellow,
                    back_colour = blue)
```

(b) Acceleration Detector along with Direction Detection using Accelerometer :

① check whether the value of z is greater than 0.975 (any axis with the required value limit).
② If yes, show the message in LED matrix, control the scroll speed and choose suitable text and background colours.
③ Print the same message in shell and clear the display.

→ Python Code -

```
if (raw["z"] > 0.975):
    sense.show_message ("Z-Axis Limit Crossed!",
                        scroll_speed = 0.005,
                        text_colour = green,
                        back_colour = red)
    print ("Z-Axis Limit Crossed!")
sense.clear() # No arguments defaults to OFF
```

(c) Assignment : To implement a multiparameter display device which will display the current temperature, humidity and pressure on the LED matrix in SenseHAT when the middle button of the joystick is pressed.

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 check if the direction is "middle" and action is "held".

③ If yes, display the current pressure, relative humidity and temperature on the LED matrix in SenseHAT.
④ Controll the scroll speed : Greater speed implies slower scroll time.
⑤ Choose suitable text and background colours.
⑥ Clear the display in SenseHAT.

→ Python code -

```
while True:
    for event in sense.stick.get_events():
        if (event.direction == "middle" and event.action == "held"):
            sense.show_message("Current Pressure : {}
                                Relative Humidity : {}
                                Temperature : {}".
            format(pressure, humidity, temperature),
                                scroll_speed = 0.005,
                                text_colour = yellow,
                                back_colour = blue)
    sense.clear()
    break
```

j) Datalogger for weather station using SenseHAT :

① Gets the current pressure in Millibars from the pressure sensor.
② Gets the percentage of relative Humidity from the humidity sensor.
③ Return the time in seconds since the epoch as a floating point number
④ Calls set_imu_config to disable the magnetometer and gyroscope. Then, gets the current orientation from the accelerometer only.
⑤ Gets the raw x, y and z axis accelerometer data.
⑥ Print the message in LED matrix and append to the log file.

→ Python Code -

```
with open("weather", "w") as f:
    while True:
```

PTO

```
humidity = sense.get_humidity()
pressure = sense.get_pressure()
temperature = sense.get_temperature()
seconds = time.time()
local_time = time.ctime(seconds)

print_message = str("Pressure: " + str(pressure) +
                    "Temperature: " + str(temperature) +
                    "Humidity: " + str(humidity) +
                    "Time: " + str(local_time))

sense.show_message(print_message,
                   scroll_speed=0.005,
                   text_colour=green,
                   back_colour="red")

f.writelines(print_message)
f.write("\n")
sense.clear() # No argument defaults to OFF
```

* **RESULT:**

Thus, performed sensor interfacing and data logging using Raspberry Pi. All the simulation results were verified successfully.

10/11/22