ESAP Lab 8

Sensehat - LED Graphics and joystick

Contents

1.1 Introduction & Setup 2

1.2 Exercises 2

LED Matrix + Joystick 2

Challenge 1 3

Challenge 2 4

Challenge 3 5

Challenge 4 6

1.1 Introduction & Setup

This lab builds on lab 7 and introduces the operation of the joystick and LED grid of the sensehat. Grading for this lab is based on completion of the 4 challenges. The challenges are composed of partially complete code that must be completed during the lab.

1.2 Exercises

LED Matrix + Joystick

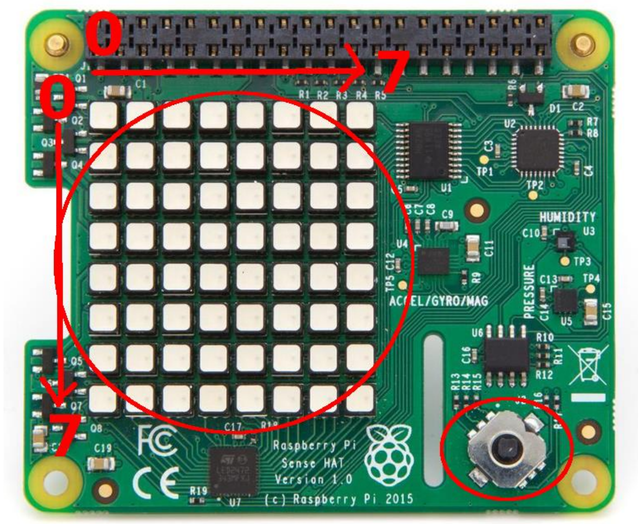
Read the online API resources for the Sensehat at: https://pythonhosted.org/sense-hat/api/#led-matrix

Before we start…

We need to import the libraries associated to Sensehat and initialize an object which will contain them:

```
from sense_hat import SenseHat

sense = SenseHat()
```



The Sensehat LED matrix is made up of 8x8 RGB LEDs, which means they can emit any colour.

The values can be changed using the syntax

```
sense.set_pixels(x, y, r, g, b)
```

where x and y correspond to the width and length of the Sensehat matrix and (r, g, b) stand for the Red, Green and Blue values (from 0 to 255) attributed to each pixel.

Other useful commands:

get_pixel() and get_pixels()
set_pixels()
clear()

Challenge 1

Insert commands in the following code such that the Sensehat displays an abstract object (smiley, circle, etc.)
Consider using several colours

```
from sense_hat import SenseHat
from time import sleep

def clear_leds(sense):
   ##
              ## Reset the values of the sensehat to 0
   ##

sense = SenseHat()
clear_leds(sense)

sense.set_pixels

## Insert code HERE ##

## Commands for image

## Added code ends HERE ##

#sense.flip_v()
angle=0
while True:
   if angle>270:
      angle=0
   sense.set_rotation(angle)
   sleep(1)
   angle+=90
```

Challenge 2

Read an image (already resized to 8x8 pixels) and display it on the Sensehat

```
from sense_hat import SenseHat
from time import sleep
from PIL import Image
import numpy as np
#import scipy.io

def load_image(values,sense):
   ##    ##
          ## Insert code Here ##
   ##    ##
def clear_leds(sense):
   ##
          ## Reset the values of the sensehat to 0
   ##

path_to_file = "Link to image file"

im = Image.open(path_to_file)
rgb_im=im.convert('RGB')
values=np.array(rgb_im)

sense=SenseHat()

clear_leds(sense)

load_image(values,sense)

sense.flip_v()
sense.set_rotation(270)
```

Challenge 3

Move a white pixel in the LED matrix, trying to reach a ‘goal pixel’ coloured green. Make sure you avoid the red pixels that are generated randomly.
Make sure that the ‘goal pixel’ is randomly generated.

```
from sense_hat import SenseHat
import numpy as np
from time import sleep

def add_dangerous_pixels(sense,nr_dang_pix):
    dangerous_coord = [] # list where all the x,y coordinates are stored
    for i in range(0,nr_dang_pix):
        ##
                    ## randomly choose x,y coordinates for the dangerous pixels
        ##
            dangerous_coord.append(candidate_pixel)
    return dangerous_coord

def check_boundary(x,y):
    ##
                    ## Add code to check if x,y are going beyond the limits of the matrix
   ##
    return x,y

sense = SenseHat()
sense.clear()

number_of_dangerous_pixels = 4
dangerous_coord = add_dangerous_pixels(sense, number_of_dangerous_pixels)

## Adding dangerous pixels to the LED matrix:
for set in dangerous_coord:
    print (set)
    sense.set_pixel(set[0],set[1],r)

white=(255,255,255)

x,y=6,6 #  location where the moving pixel is being ‘spawned’
sense.set_pixel(x,y,white)

## Set wining pixel
goal_pixel=(2,2)
##
                ## Add code to generate “objective pixel”
##
sense.set_pixel(goal_pixel[0],goal_pixel[1],0,255,0)

alive=True # flag to check where the moving pixel is located
```

```
while alive:
    for event in sense.stick.get_events():

        if event.direction=='up':
            sense.set_pixel(x,y,0,0,0)

            y-=1
            x,y=check_boundary(x,y)
            sense.set_pixel(x,y,white)

        if event.direction=='down':
            sense.set_pixel(x,y,0,0,0)

            y+=1
            x,y=check_boundary(x,y)
            sense.set_pixel(x,y,white)

        if event.direction=='left':
            sense.set_pixel(x,y,0,0,0)

            x-=1
            x,y=check_boundary(x,y)
            sense.set_pixel(x,y,white)

        if event.direction=='right':
            sense.set_pixel(x,y,0,0,0)

            x+=1
            x,y=check_boundary(x,y)
            sense.set_pixel(x,y,white)

        if (x,y) in dangerous_coord:
            sense.show_message('Game over')
            print('Game Over')
            alive=False # Game is over, because you lost
        if (x,y) == (goal_pixel[0],goal_pixel[1]):
            sense.show_message('Victory')
            alive=False # Game is over, because you won.
```

Challenge 4

Use the accelerator to build an “earthquake detector”.

Use the 8x8 grid to flash red lights for a “severe earthquake”, yellow lights for a “mild earthquake” and a solid green light for negligible detected acceleration.

Continue flashing “red” lights for at least 10 seconds after a “severe” event and at least 5 second after a “mild” event. If a “severe” event occurs while a yellow event is ongoing, the red event must be displayed immediately.

As we will not have time to wait until Galway experiences an actual earthquake to test your code, you may define the thresholds for “negligible, mild, and severe” in such a way that allows you to produce these effects on demand for demonstration to the TA.