Massey University
School of Engineering and Advanced Technology
Computer Science and Information Technology

Discover Computing With the Raspberry Pi 3

Activity 1: Programming Artistic Graphics on the Raspberry Pi 3

Description and Aim of Activity:

In this activity, you will learn how to manipulate colours of pixels on the LED matrix of the Sense HAT on the Raspberry Pi 3 and use what you have learnt to solve the challenge of writing or modifying a Python program that draws a decorated Christmas tree.

Learning Objectives:

At the end of this activity, you should be able to:

1. Describe the coordinate system used by most computer graphics systems;
2. Specify the address of any pixel on the LED matrix by using the graphics coordinate notation;
3. Describe how colours are specified by using the RBG decimal notation;
4. Programmatically manipulate the colour of a pixel on the LED matrix in the Python programming language;
5. Write a Python program that runs on the Raspberry Pi 3 to display some art, e.g., a decorated Christmas tree, on the Sense HAT's LED matrix

Specifying pixel coordinates and manipulating pixels on the LED matrix of the Sense HAT

x-axis

y-axis

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5							(6,5)	
6								
7								

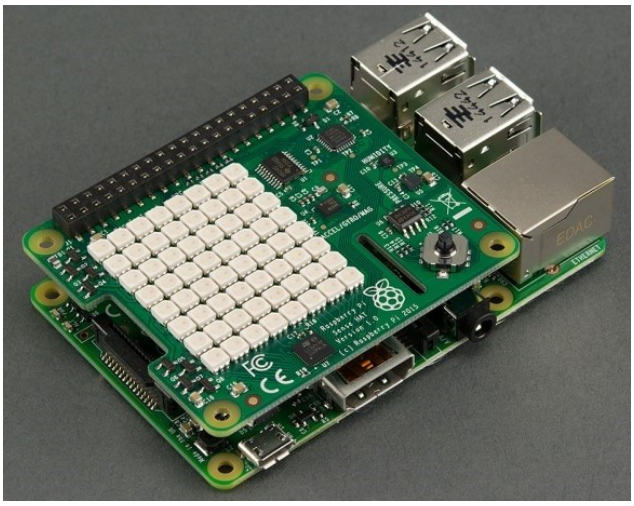
A pixel's location is specified by the coordinates **`(x,y)`** on the grid illustrated on the left.

The pixel whose location is `(6,5)` is indicated on the grid.

The coordinates for the LED matrix on the Sense HAT.

0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0	7, 0
0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1	7, 1
0, 2	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2	7, 2
0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3	7, 3
0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	6, 4	7, 4
0, 5	1, 5	2, 5	3, 5	4, 5	5, 5	6, 5	7, 5
0, 6	1, 6	2, 6	3, 6	4, 6	5, 6	6, 6	7, 6
0, 7	1, 7	2, 7	3, 7	4, 7	5, 7	6, 7	7, 7

Pixel coordinates on the LED matrix. See Appendix B for a larger table of coordinates



LED matrix on the Sense HAT

On the grid, each pixel can be assigned a colour, which is specified in the Python program as a list of base colours, `[r, b, g]` where each `r, b` and `g` is a number in the range `0 to 255`. The RBG colour specification is equivalent to mixing the based colour red, blue and green to come up with almost any color.

To set the colour to `green` for the pixel at `(6,5)` in Python, the program is written as follows:

```
from sense_hat import SenseHat
sense = SenseHat()
x=6
y=5
sense.set_pixel(x,y,34,139,34)
colour = sense.get_pixel(x,y)
print(colour)
```

Note that, in the above program, the green colour is specified by using the RBG (red, blue, green) format as `34,139,34.`

Note also that you can get the colour of a pixel at `(x,y)` by calling the function, `sense.get_pixel(x,y).` You will notice that the pixel values you pass into set_pixels sometimes change when you read them back with get_pixels. This is because we specify each pixel element as 8 bit numbers (0 to 255) but when they're passed into the Linux frame buffer for the LED matrix the numbers are bit shifted down to fit into RGB 565. 5 bits for red, 6 bits for green and 5 bits for blue. The loss of binary precision when performing this conversion (3 bits lost for red, 2 for green and 3 for blue) accounts for the discrepancies you see.

Drawing the Christmas tree on the LED matrix

The following grid specifies the undecorated Christmas tree.

	0	1	2	3	4	5	6	7
0	o	o	o	g	g	o	o	o
1	o	o	g	g	g	g	o	o
2	o	o	g	g	g	g	o	o
3	o	g	g	g	g	g	g	o
4	o	g	g	g	g	g	g	o
5	g	g	g	g	g	g	g	g
6	g	g	g	g	g	g	g	g
7	o	o	b	b	b	b	o	o

Where:

`g = [34,139,34]`, which is the **green** colour,

`b = [139,69,19]`, which is the **brown** colour,

`o = [0, 0, 0]`, which is the **black** colour and the LED light will be switched off.

The undecorated Christmas Tree displayedon the LED matrix of the Sense HAT

In the Appendix, you are provided with blank grids that you can use to design your own artwork before writing the Python program that would draw the artwork on the LED matrix.

Activity 1.1: Run the given program to draw the undecorated Christmas Tree

In IDLE3, open and run the program: `/home/pi/rpi/led-xmas-tree1.py`

For explanation of the program code study the comments inside the program. This is important in order to prepare for the next two challenges.

Activity 1.2: Challenge 1 – Decorate the Christmas Tree

Open the program, `/home/pi/rpi/led-xmas-tree2.py`, in IDLE 3.

Modify the program so that it would draw a decorated Christmas tree. At the top of the tree, make the two pixel white to represent the white star used to decorate the top of most Christmas trees.

Hint: To add decorations, all you need to do is to select green pixels that you would like to turn into decorations by simply changing the colour to another colour for decorating the tree.

Activity 1.3: Challenge 2 – Simulate Christmas Tree Decorations Lights With Random Colours

Open the program, `/home/pi/rpi/led-xmas-tree3.py`, in IDLE 3.

Modify the program so that each decoration pixel *(except the white pixels at the top)* displays a random colour every second. The white pixels at the top should toggle between white and black/off.

Appendix A

Blank 8 x 8 Grids for Use to Design Graphics

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								

Appendix B:

Coordinates for the 8 x 8 LED Matrix

0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0	7, 0
0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1	7, 1
0, 2	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2	7, 2
0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3	7, 3
0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	6, 4	7, 4
0, 5	1, 5	2, 5	3, 5	4, 5	5, 5	6, 5	7, 5
0, 6	1, 6	2, 6	3, 6	4, 6	5, 6	6, 6	7, 6
0, 7	1, 7	2, 7	3, 7	4, 7	5, 7	6, 7	7, 7