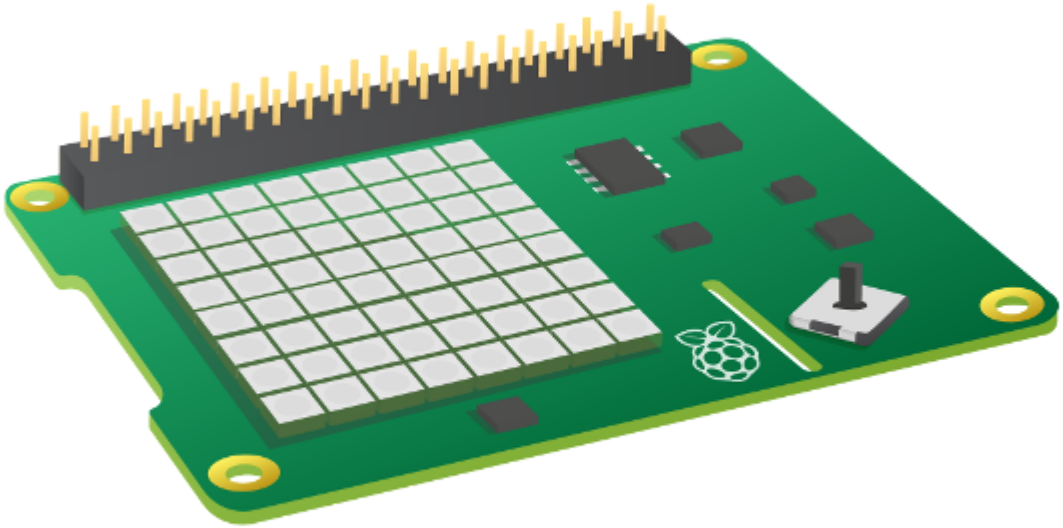
Getting started with the Sense HAT

Introduction

What you will make

The Sense HAT, which is a fundamental part of the Astro Pi mission, allows your Raspberry Pi to sense the world around it.



In this project, you will learn how to control the Sense HAT's LED matrix and collect sensor data, and you will combine these ideas in a number of small projects.

What you will learn

By following this resource with your Raspberry Pi and Sense HAT you will learn how to:

- Communicate with the Sense HAT using Python
- Access the outputs of the Sense HAT

- Program the inputs of the Sense HAT
- Use the Sense HAT library to display messages and images
- Use variables to store sensor data
- Use loops to repeat behaviours

This resource covers elements from the following strands of the Raspberry Pi Digital Making Curriculum:

- Use basic programming constructs to create simple programs
- Process input data to monitor or react to the environment

What you will need

Hardware

- Raspberry Pi
- Sense HAT

Software

You will need the latest version of Raspbian which already includes the following software packages:

- Python 3
- Sense HAT for Python 3

If for any reason you need to install a package manually, follow these instructions:

Install a software package on the Raspberry Pi

Your Raspberry Pi will need to be online to install packages. Before installing a package, update and upgrade Raspbian, your Raspberry Pi's operating system.

- Open a terminal window and enter the following commands to do this:

```
sudo apt-get update
sudo apt-get upgrade
```

- Now you can install the packages you'll need by typing `install` commands into the terminal window. For example, here's how to install the Sense HAT software:

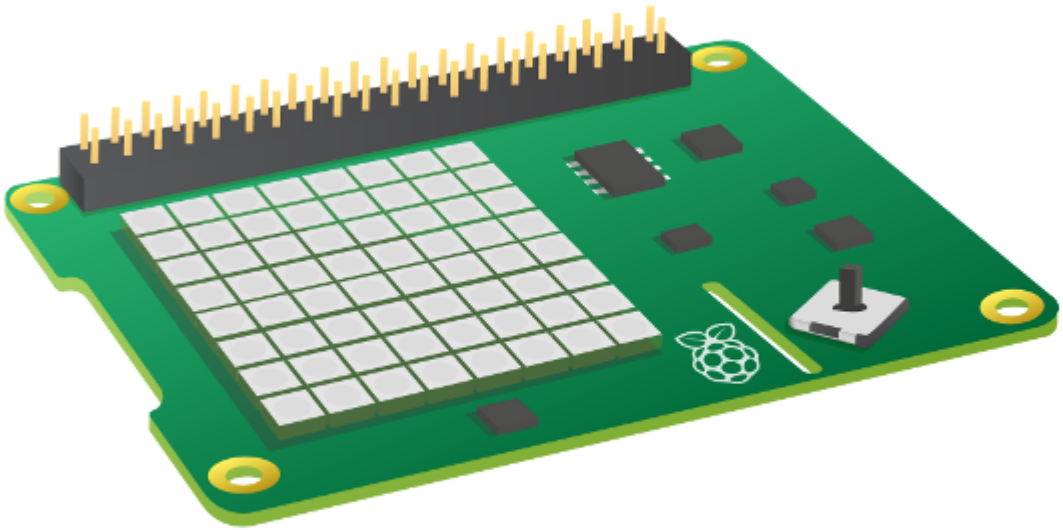
```
sudo apt-get install sense-hat
```

Type this command into the terminal to install the Sense HAT package:

```
sudo apt-get install sense-hat
```

What is a Sense HAT?

The Sense HAT is an add-on board for the Raspberry Pi, made especially for the Astro Pi competition. The board allows you to make measurements of temperature, humidity, pressure, and orientation, and to output information using its built-in LED matrix.



Attaching a Sense HAT

Before attaching any HAT to your Raspberry Pi, ensure that the Pi is shut down.

- Remove the Sense HAT and parts from their packaging.

- Use two of the provided screws to attach the spacers to your Raspberry Pi, as shown below.
- Then push the Sense HAT carefully onto the pins of your Raspberry Pi, and secure it with the remaining screws.

NOTE: Using a metal stand-off next to the Raspberry Pi 3's wireless antenna will degrade its performance and range. Either leave out this stand-off, or use nylon stand-offs and nylon screws instead.

If you don't have access to a real Sense HAT, you can use an emulator.

Using the Sense HAT emulator

If you don't have access to a Sense HAT, you can use the emulator.

Online Sense HAT emulator

There is an online emulator you can use in your browser to write and test code for the Sense HAT.

- Open an internet browser and go to https://trinket.io/sense-hat.
- If you would like to save your work, you will need to create a free account on the Trinket website.

Sense HAT emulator on the Raspberry Pi

If you are using a Raspberry Pi, there is a Sense HAT emulator included in the Raspbian operating system.

- From the main menu, select **Programming** > **Sense HAT emulator** to open a window containing the emulator.
- If you are using this version of the emulator, your program must import from `sense_emu` instead of `sense_hat`:

```
from sense_emu import SenseHat
```

If you later want to run your code on a real Sense HAT, just change the import line as shown below. All other code can remain exactly the same.

```
from sense_hat import SenseHat
```

Displaying text

- Display the text “Astro Pi is awesome” on your Sense HAT’s LED display.

Show a message on the Sense HAT

Make sure you have the following lines of code in your Python program to set up your connection with the Sense HAT. There is no need to add them more than once.

```
from sense_hat import SenseHat
sense = SenseHat()
```

- Add this code to display a message on the Sense HAT’s LED matrix.

```
sense.show_message("Hello world")
```

The message “Hello world” will now scroll across the LED screen.

- Change the words in the quotes (`""`) to see a different message.

You can try it out here:

We can change how the message is displayed by adding some extra **parameters** to the `show_message` command.

`scroll_speed`: affects how quickly the text moves across the screen. The default value is `0.1`. The bigger the number, the lower the speed.

`text_colour`: alters the colour of the text and is defined via three values to specify red, green, and blue. These are also called RGB values.

Check out the sections below to learn more about RGB values.

Displaying a colour on the Sense HAT

- In a Python file, type in the following code:

```
from sense_hat import SenseHat

sense = SenseHat()

r = 255
g = 255
b = 255
```

```
sense.clear((r, g, b))
```

- Save and run your code. The LED matrix will then go bright white.
- The variables `r`, `g`, and `b` represent the colours red, green, and blue. Their values specify how bright each colour should be; each value can be between `0` and `255`. In the above code, the maximum value for each colour has been used, so the result is white.
- You can also define all three RGB values of a colour using a single line of code:

```
red = (255,0,0)
```

- Change the value of one of the colours, then run the code again. What do you see?
- Which other colours can you make?

Representing colours with numbers

The colour of an object depends on the colour of the light that it reflects or emits. Light can have different wavelengths, and the colour of light depends on the wavelength it has. The colour of light according to its

wavelength can be seen in the diagram below. You might recognise this as the colours of the rainbow.

Humans see colour because of special cells in our eyes. These cells are called *cones*. We have three types of cone cells, and each type detects either red, blue, or green light. Therefore all the colours that we see are just mixtures of the colours red, blue, and green.

In additive colour mixing, three colours (red, green, and blue) are used to make other colours. In the image above, there are three spotlights of equal brightness, one for each colour. In the absence of any colour the result is black. If all three colours are mixed, the result is white. When red and green combine, the result is yellow. When red and blue combine, the result is magenta. When blue and green combine, the result is cyan. It's possible to make even more colours than this by varying the brightness of the three original colours used.

Computers store everything as 1s and 0s. These 1s and 0s are often organised into sets of 8, called **bytes**.

A single byte can represent any number from 0 up to 255.

When we want to represent a colour in a computer program, we can do this by defining the amounts of red, blue, and green that make up that colour. These amounts are usually stored as a single byte and therefore as a number between 0 and 255.

Here's a table showing some colour values:

Red	Green	Blue	Colour
255	0	0	Red
0	255	0	Green
0	0	255	Blue
255	255	0	Yellow
255	0	255	Magenta
0	255	255	Cyan

You can find a nice colour picker to play with at w3schools.

`back_colour`: alters the colour of the background and works in the same way as `text_colour`.

- Add a line of code before your message to define a variable called `blue` with the value `(0, 0, 255)`.

Creating a variable in Python

A variable allows you to store data within a program. Variables have a **name** and a **value**.

This variable has the name `animal` and the value `cat`:

```
animal = "cat"
```

This variable has the name `score` and the value `30`:

```
score = 30
```

To create a variable, give it a name and set it equal to a value. The name of the variable always goes on the left, so this code is wrong:

```
# This code is wrong
30 = score
```

- Add another line of code to define a variable called `yellow` with the value `(255, 255, 0)`.
- Add parameters to the `show_message` command to display the text in yellow with a blue background.

Solution

The part to add is highlighted in blue.

- Add another parameter called `scroll_speed` to the `show_message` command and set the speed equal to `0.05` to speed up how quickly your message scrolls.
- Put your scrolling message in a while loop to make it repeat.

While True loop in Python

The purpose of a **while** loop is to repeat code over and over while a condition is `True`. This is why while loops are sometimes referred to as **condition-controlled** loops.

The example below is a while loop that will run forever - an infinite loop. The loop will run forever because the condition is always `True`.

```
while True:
    print("Hello world")
```

Note: The `while` line states the loop **condition**. The `print` line of code below it is slightly further to the right. This is called **indentation** - the line is indented to show that it is inside the loop. Any code inside the loop will be repeated.

An infinite loop is useful in situations where you want to perform the same actions over and over again, for example checking the value of a sensor. An infinite loop like this will **block** - this means that any lines of code written after the loop will never happen.

Solution

Displaying a single character

- Display the letter “A” on your Sense HAT’s LED display.

Show a letter on the Sense HAT

Make sure you have the following lines of code in your program to set up your connection with the Sense HAT. There is no need to add the code more than once.

```
from sense_hat import SenseHat
sense = SenseHat()
```

- Add this code to display a single letter on the LED matrix:

```
sense.show_letter("Z")
```

If you run this code, the letter “Z” will appear on the screen. You can change which letter is displayed by altering the letter in the quote marks (`""`).

Try it out here:

We can change how the letter is displayed by using two of the same parameters we used for the `show_message` command: `text_colour` and `back_colour`. Letters do not scroll, so there is no `scroll_speed` parameter.

- Display the letter “J” in red on a white background.
- Use the `sleep` function to display the letters of your name one at a time, each in a different colour, with a one-second pause between each.

Using Python's sleep command

You can use the `sleep` function to temporarily pause your Python program.

- Add this line of code at the top of your program to import the `sleep` function.

```
from time import sleep
```

- Whenever you want a pause in your program, call the `sleep` function. The number in the brackets indicates how many seconds you would like the pause to be.

```
sleep(2)
```

You can pause for fractions of a second as well.

```
sleep(0.5)
```

Solution

- Randomly generate a colour by using `randint` to choose a number between `0` and `255` for each of the three RGB values that make up a colour.

Randomness in Python

One of the standard modules in Python is the `random` module. You can use it to create pseudo-random numbers in your code.

randint

You can generate random integers between two values using the `randint` function. For example, the following line of code will produce a random integer between 0 and 10 (inclusive).

```
from random import randint
num = randint(0,10)
```

uniform

If you want a random floating-point number (also called float), you can use the `uniform` function. For example, the following line of code will produce a random float that's equal to or greater than 0, but less than 10.

```
from random import uniform
num = uniform(0,10)
```

choice

If you want to choose a random item from a list, you can use the `choice` function.

```
from random import choice
deck = ['Ace', 'King', 'Queen', 'Jack']
card = choice(deck)
```

Solution

- Use `sense.clear()` at the end of your code to clear the LED matrix.

Displaying images

You can fill the whole LED matrix with a single colour by using the `clear` method with the colour you've picked.

Displaying a colour on the Sense HAT

- In a Python file, type in the following code:

```
from sense_hat import SenseHat

sense = SenseHat()

r = 255
g = 255
b = 255
```

```
sense.clear((r, g, b))
```

- Save and run your code. The LED matrix will then go bright white.
- The variables `r`, `g`, and `b` represent the colours red, green, and blue. Their values specify how bright each colour should be; each value can be between `0` and `255`. In the above code, the maximum value for each colour has been used, so the result is white.
- You can also define all three RGB values of a colour using a single line of code:

```
red = (255,0,0)
```

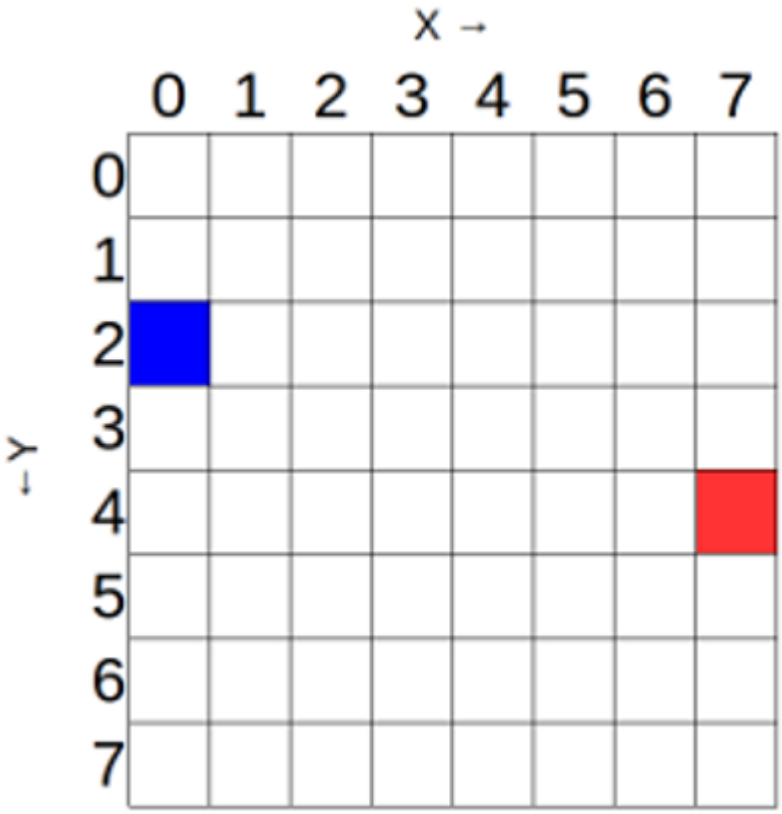
- Change the value of one of the colours, then run the code again. What do you see?
- Which other colours can you make?

Setting single pixels

The LED matrix can display more than just text! We can control each LED individually to create an image.

Sense HAT LED matrix coordinates

The Sense HAT's LED matrix uses a coordinate system with an x- and a y-axis. The numbering of both axes begins at `0` (not 1) in the top left-hand corner. Each LED can be used as one pixel of an image, and it can be addressed using an `x, y` notation.



The blue pixel is at coordinates `0, 2`.
The red pixel is at coordinates `7, 4`.

You can set pixels (LEDs) individually using the `set_pixel()` method.

To replicate the diagram above, you would enter a program like this:

- Light up the pixels (LEDs) in the four corners of the matrix in a colour of your choice.

Can you guess what this code creates?

- Change the code to make a different pixel picture.

Setting multiple pixels

Setting pixels individually can work brilliantly, but it gets rather complex when you want to set multiple pixels. To change all the pixels in one go with the `set_pixels` command.

- Use the `set_pixels` method to display an image on the LED matrix.

Setting multiple pixels on the Sense HAT

You may be tempted to try to draw shapes on the Sense HAT's LED matrix by using the `set_pixel` command over and over. However, there is a `set_pixels` command with which you can change all 64 LEDs using one single line of code!
For example, you could draw a Minecraft creeper face on the LED Matrix:

You can even use more than two colours, like in this example of Steve from Minecraft:

Setting orientation

So far, all our text and images have appeared the same way up, with the HDMI port at the bottom. However, this may not always be the true orientation of the Sense HAT (especially in outer space), so sometimes you might want to change the orientation of the LED matrix.

Rotate the Sense HAT's LED display

You can change the orientation of the LED matrix display on the Sense HAT. Use the `set_rotation()` method to rotate the screen one of four ways: by 0, 90, 180, or 270 degrees.

- For example, to rotate your display by 180 degrees you'd use this code:

```
sense.set_rotation(180)
```

- You can also flip the image on the screen, either horizontally…

```
sense.flip_h()
```

…or vertically.

```
sense.flip_v()
```

- Change the orientation of the pixel picture from the previous step by adding code to rotate the display below the code you wrote to connect to the Sense HAT:

You can create a simple animation by flipping an image repeatedly:

Sensing the environment

The Sense HAT has a set of environmental sensors for detecting the surrounding conditions; it can measure pressure, temperature, and humidity.

Reading pressure with the Sense HAT

- In a Python file, enter the following code:

```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

pressure = sense.get_pressure()
print(pressure)
```

- When you run the program, you should see something like this:

```
1013.40380859
```

Detecting temperature with the Sense HAT

The Sense HAT has two sensors capable of reading the ambient temperature: the humidity sensor and the pressure sensor. `get_temperature_from_humidity` reads the temperature from the humidity sensor (`get_temperature` is a short version of the same command). `get_temperature_from_pressure` reads the temperature from the pressure sensor.

- In a Python file, enter the following code:

```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

temp = sense.get_temperature()
print(temp)
```

- You should see something like this:

```
28.6293258667
```

Detecting humidity with the Sense HAT

- In a Python file, enter the following code:

```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

humidity = sense.get_humidity()
print(humidity)
```

- When you run the program, you should see something like this:

```
34.6234588623
```

- Create a scrolling text display which keeps people informed about the current pressure, temperature, and humidity readings. You can use the scrolling text display code you wrote in the ‘Displaying text’ step to help you.

Solution

According to online documentation, the International Space Station maintains these conditions at the following levels:

Temperature: 18.3-26.7 Celsius

Pressure: 979-1027 millibars

Humidity: around 60%

- Define variables for the colours green (0, 255, 0) and red (255, 0, 0).
- Use an if statement in your code to check whether the temperature is between 18.3 and 26.7 degrees Celsius.

Boolean operators in Python conditionals

- A standard `if` statement in Python checks for a single condition. For instance:

```
x = 5
if x > 0:
```

```
        print('x is greater than zero')
```

- But sometimes you might want to check for more than one condition. In such cases, you can use logical operators in your code.
- The `and` operator checks to see if two conditions have both been met. For instance:

```
x = 5
if x > 0 and x < 10:
    print('x is between 0 and 10')
```

- So long as `x` is any number within the group - `1,2,3,4,5,6,7,8,9`, then the condition will be true.
- You can use the `or` operator to check if either of the conditions is true.

```
x = 5
if x > 0 or x < 10:
        print('x exists')
```

- In this case, the condition will be true as long as `x` is greater than `0`, or if it is less than `10`.
- If the temperature is within this normal range, display the scrolling message with a green background. If not, display a red background.

Solution

- Add more if statements to test for normal pressure and humidity conditions as well.

Detecting movement

The Sense HAT has an IMU (**I**nertial **M**easurement **U**nit) chip which includes a set of sensors that detect movement:

- A gyroscope (for detecting which way up the board is)
- An accelerometer (for detecting movement)
- A magnetometer (for detecting magnetic fields)

What is an IMU?

The Sense HAT has a movement sensor called an IMU, which measures the kinds of movement it experiences. IMU stands for Inertial Measurement Unit. It's actually three sensors in one:

- A gyroscope: measures momentum and rotation
- An accelerometer: measures acceleration forces, can be used to find the direction of gravity

- A magnetometer: measures the Earth's own magnetic field, a bit like a compass

Why is a movement sensor important? When you're in space, there is one question of absolute importance to which you must always know the answer: "Which way am I pointing?"

If you don't know your orientation, you are in big trouble. So an IMU sensor like the one of the Sense HAT is used on all manned and unmanned spacecraft to track movements and maintain an understanding of orientation. Even the earliest spacecraft had them — ask your grandparents if they remember the Apollo mission that landed humans on the surface of the moon.

Above is a picture of the IMU sensor from the Apollo command module. You'll notice how big it is compared to the tiny black cube on the Astro Pi — that's the difference between 1975 and 2015 technology. Incidentally, the Astro Pi IMU is probably not as accurate as the Apollo one; however, it is a million times cheaper!

Learn about pitch, roll, and yaw

We all know the Earth rotates around an axis that runs between the North and South Poles. All objects have **three axes** around which they can rotate. These are:

- **Pitch** — imagine a plane taking off
- **Roll** — imagine a plane doing a victory roll
- **Yaw** — imagine steering a plane like a car

If you know how much rotation has happened on each axis of an object, then you know which way the object is pointing.

Watch this short video showing where these axes are in relation to a plane. Try to imagine a plane pointing in a random direction in space. To get the plane into that position, you can rotate it by a known amount around each of the three axes.

The image below shows where the three axes are in relation to the Sense HAT.

- Write a program to detect the current pitch, roll, and yaw. Run the program and move the Sense HAT around. Watch how the values change as the Sense HAT moves.

Detecting pitch, roll, and yaw with the Sense HAT

The Sense HAT has orientation sensors which detect pitch, roll, and yaw. Do the following to access these data.

- In a Python file, enter this code:

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()

o = sense.get_orientation()
pitch = o["pitch"]
roll = o["roll"]
yaw = o["yaw"]
print("pitch {0} roll {1} yaw {2}".format(pitch, roll, yaw))
```

- When you run the program, you should see something like this:

```
pitch 356.35723002363454 roll 303.4986602798494 yaw 339.19880231669873
```

Note: When using the movement sensors, it is important to take frequent readings. If you take readings too slowly, for example by putting `time.sleep(0.5)` in your loop, you will see strange results. This is because the code needs lots of measurements in order to successfully combine the data coming from the gyroscope, accelerometer, and magnetometer.

Which way up?

The `sense.get_accelerometer_raw()` method tells you the amount of G-force acting on each axis (x, y, z). If any axis has ±1G, then you know that axis is pointing downwards.

In this example, the amount of gravitational acceleration for each axis is extracted and is then rounded to the nearest whole number:

- Rotate the Sense HAT. You should see the values for `x` and `y` change between `-1` and `1`. If you place the Pi flat or turn it upside down, the value for the `z` axis will be `1` and then `-1`.

Use this information to set the orientation of the LED matrix.

- Starting with the code above, add some code before the while loop to display the letter "J" on the LED matrix. Use the `show_letter` method you already learned about.

- After the code which displays the G-force values for the x, y and z axes , add an if statement to check which way up the Sense HAT is pointing. Update the orientation of the display using the `set_rotation` method you learned about earlier. Here is some pseudo-code to get you started:

```
If the x axis has -1 G, rotate 180 degrees
Else if the y axis has 1 G, rotate 90 degrees
Else if the y axis has -1 G, rotate 270 degrees
Else rotate 0 degrees
```

Solution

Shake the board

If the board is only rotated, it will only ever experience 1 G of acceleration in any direction; if we were to shake it, the sensor would experience more than 1 G. We could then detect that rapid motion and respond.

For this program we will introduce the `abs()` function, which is not specific to the Sense HAT library but instead is part of standard Python. `abs()` gives us the absolute figure of a value and ignores whether the actual value is positive or negative — for example, `abs(1)` and `abs(-1)` both return `1`. This function is helpful because we don't care in which direction the sensor is being shaken, just that it is shaken.

```
from sense_hat import SenseHat

sense = SenseHat()

red = (255, 0, 0)

while True:
    acceleration = sense.get_accelerometer_raw()
    x = acceleration['x']
        y = acceleration['y']
        z = acceleration['z']
```

```
x = abs(x)
y = abs(y)
z = abs(z)

if x > 1 or y > 1 or z > 1:
    sense.show_letter("!", red)
else:
    sense.clear()
```

This is a little tricky to emulate, so you should try this one using a real Sense HAT if you can. If you find the program is too sensitive (that is, it thinks the Sense HAT is constantly being shaken), try changing the value `1` to a larger value to raise the threshold of what is defined as a "shake".

Using the joystick

You can detect when the Sense HAT's joystick is pressed, held, and released in five different directions: up, down, left, right, and middle.

Detecting joystick movement with the Sense HAT

The Sense HAT joystick is mapped to the four keyboard cursor keys, and the joystick's middle-click is mapped to the `Return` key. This means that using the joystick has exactly the same effect as pressing those keys on the keyboard. Remember that the down direction is with the HDMI port facing downwards.

- In a Python file, enter the following code:

```
from sense_hat import SenseHat
sense = SenseHat()


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

This code will print out the direction that the joystick is pushed in, or the direction from which it was released.

- When you run the program and move the joystick in various directions, you should see something like the following in the terminal window. In the trinket emulator, you can pretend to move the joystick using your keyboard's cursor keys.

```
('up', 'pressed')
('up', 'released')
('down', 'pressed')
('down', 'released')
('left', 'pressed')
('left', 'released')
('right', 'pressed')
('right', 'released')
('middle', 'pressed')
('middle', 'released')
```

- Depending on which way the joystick was pressed, display one of the letters U, D, L, R or M on the LED matrix.

Solution

You can also call a function whenever the Sense HAT's joystick is moved in a particular direction.

Triggering function calls with the Sense HAT joystick

The Sense HAT joystick can be used to trigger function calls in response to being moved.

- For instance, you can tell your program to continually 'listen' for a specific event, such as the joystick being pushed up (`direction_up`), and to then trigger a function (called `pushed_up` in this example) in response.

```
sense.stick.direction_up = pushed_up
```

- The function triggered by the event can either have no parameters, or it can take the event as a parameter. In the example below, the event is simply printed out.

```
def pushed_up(event):
    print(event)
```

- This function would print a timestamp of the event, the direction in which the joystick was moved, and the specific action. The output would look similar to this:

```
InputEvent(timestamp=1503565327.399252, direction=u'up', action=u'pressed')
```

- Another useful example is the `direction_any` method:

```
sense.stick.direction_any = do_thing
```

- If you use this method as seen in the example, the `do_thing` function will be triggered in response to any joystick event. For instance, you could define the `do_thing` function so that it reports the exact event in plain English.

```
def do_thing(event):
    if event.action == 'pressed':
        print('You pressed me')
        if event.direction == 'up':
            print('Up')
        elif event.direction == 'down':
            print('Down')
    elif event.action == 'released':
        print('You released me')
```

- Create functions to fill the LED matrix with four different colours. Add triggers to call one function for each possible direction in which the joystick can be pressed.

Solution

Challenge: putting it all together

Now that you've explored most of the features of the Sense HAT, you can combine them to create a project. Here's an example reaction game, which could be used by the astronauts to test their reflexes:

Rotate the board to make the arrow point up. If you match it in time, the arrow turns green and your score increases; if not, your arrow turns red and the game ends. The game keeps showing arrows in new orientations until you lose, and each turn gets faster.

This idea combines:

- Showing messages and images on the LED matrix
- Setting and detecting the orientation
- Use of variables, randomisation, iteration, and selection

As this is more complicated than previous programs in this resource, it's worth planning out the steps involved in pseudo-code:

`Import` the required libraries (`sense_hat`, `time`, `random`)
Create a sensehat object

Define variables for the colours needed (white, green, red, blank)
Create three different arrows (white, green, red)
Set a variable `pause` to 3 (the initial time between turns)

Set variables `score` and `angle` to `0`
Set a variable called `play` to `True` (this will be used to stop the game later)

`while play == True`
Choose a new `random` angle
Display the white arrow
`Sleep` for current pause length
`If` orientation matches the arrow…
Add a point and turn the arrow green
Otherwise set play to `False` and display the red arrow

Shorten the pause duration slightly
Pause before the next arrow

When loop is exited, display a message with the score

Solution

Challenge: more ideas

Now that you have explored the basics of the Sense HAT, you might want to investigate other things to do with it:

- Tell a joke on the LED screen.
- If your Sense HAT is connected to the internet, you could use a Twitter API library to make it display incoming tweets.
- Create your own images to display on the LED matrix.
- Can you alternate between images to create an animation? Check out this Geek Gurl Diaries video for some inspiration.
- Create an electronic dielike this one. Shaking the Pi triggers the roll of the die.
- Create a simple graphical thermometer which outputs different colours or patterns depending on the temperature.
- Write a program that displays an arrow (or other symbol) on screen; this symbol could be used to point to which way is down. This way, astronauts in low gravity always know where the Earth is.
- Use the accelerometer to sense small movements — this could form part of a game, alarm system, or even an earthquake sensor.
- Make use of the humidity sensor to detect breath and display a colour depending on the humidity.

Published by the Raspberry Pi Foundation – www.raspberrypi.org