Getting Started with Rainbow HAT

This tutorial will show you how to install the Rainbow HAT Python library, and then walk through its functionality.

Installing the software

We always recommend using the most up-to-date version of Raspbian, as this is what we test our boards and software against, and it often helps to start with a completely fresh install of Raspbian, although this isn't necessary.

As with most of our boards, we've created a really quick and easy one-line-installer to get your Rainbow HAT set up. We'd suggest that you use this method to install the Rainbow HAT software.

Open a new terminal, and type the following, making sure to type 'y' or 'n' when prompted:

```
curl -sS get.pimoroni.com/rainbowhat | bash
```

Once that's done, it probably a good idea to reboot your Pi to let the changes propagate.

Using the software

Open a new terminal and type `sudo python` to open a new Python prompt.

To import the Rainbow HAT library, type the following in your Python prompt that you just opened:

```python
import rainbowhat as rh
```

Notice that we said `as rh`. That just saves us from typing `rainbowhat` every time we want to use something from the library.

The rainbow

Rainbow HAT's rainbow is a single sweep of 7 APA102 RGB LED pixels that can be controlled individually through the Rainbow HAT Python library. As is the convention in Python, the seven pixels are numbered from 0 to 6, and arranged from right to left (pixel 0 is the rightmost, pixel 6 is the leftmost).

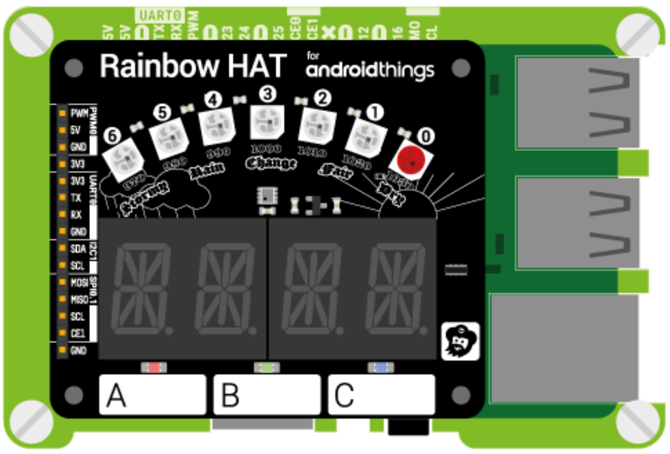
There are two methods that we can use to control the rainbow pixels - `set_pixel` and `set_all` - the former allows per-pixel control of colour, and the latter sets all of the pixels to the same colour.

We'll begin by lighting the first pixel red. The `set_pixel` function needs to be passed four pieces of information - the pixel number, and a red, green, and blue brightness value (from 0 to 255). Once pixel values have been set, you have call the `show` function to update the pixels and show the colours that you've set.

Type the following:

```python
rh.rainbow.set_pixel(0, 255, 0, 0)

rh.rainbow.show()
```

The rightmost pixel should just have lit red. Notice that we typed `rh.rainbow.set_pixel(0, 255, 0, 0)`. All of the functions related to the Rainbow are collected together into a module called `rainbow`, hence why we had to type `rh.rainbow` before the name of the function we wanted to use.



Now, let's get a little more advanced and try making our red pixel scan across from right to left. We'll use a `while`loop and a `for` loop to do this.

Type the following, pressing control-c to break out:

```
import time


while True:

    for pixel in range(7):

        rh.rainbow.clear()

        rh.rainbow.set_pixel(pixel, 255, 0, 0)

        rh.rainbow.show()

        time.sleep(0.1)
```

The red pixel should now be scanning across from right to left.

We introduced a couple of new functions here - `time.sleep(0.1)` and `rh.rainbow.clear()`. The `time.sleep(0.1)` allows us to introduce a small pause of 0.1 seconds between lighting each pixel and therefore to set the animation speed (decrease the number to speed it up, increase it to slow it down). The`rh.rainbow.clear()` clears any set pixels, and we call this function each time at the top of our `for` loop so that only a single pixel is lit at any one time.

Now, we'll use the `set_all` function to control all of the pixels at once and then make them scan through all the colours of the rainbow. Because this function controls all of the pixels we don't have to pass a pixel number in, just the RGB values.

To scan through the colours of the rainbow, we're going to use HSV colour and convert it to RGB before passing it to the `set_all` function. HSV colour is handy for this because the H of the HSV controls the hue of the colour and represents a position around the colour wheel. This means that we can just cycle around and transition between all of the colours of the rainbow.

Type the following, pressing control-c to break out:

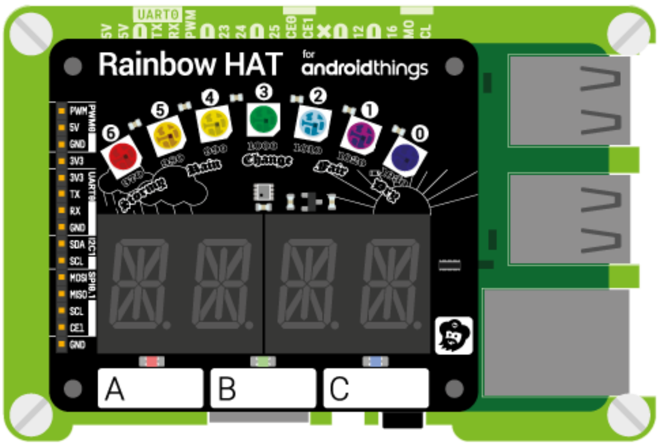
```
import colorsys
import time

rh.rainbow.clear()

while True:
    for i in range(101):
        h = i / 100.0
        r, g, b = [int(c * 255) for c in colorsys.hsv_to_rgb(h, 1.0, 1.0)]
        rh.rainbow.set_all(r, g, b)
        rh.rainbow.show()
        time.sleep(0.02)
```

This is pretty complex, so let's look at each step. First, we had to import the `colorsys` library to use its `hsv_to_rgb`function to convert our HSV colour to RGB.

Again, we're using a `while True` loop that will run continuously and then a `for` loop inside that. The `for` loop is what will cycle through the hues in the colour wheel. Conveniently, 0 and 1 are both at the same position on the wheel, so we just have to go from 0 up to 1 then start again at 0. We're just using a range of numbers from 0 to 100 and dividing them by 100 to get 101 values between 0 and 1 (where it says `h = i / 100.0`).

The line that says `r, g, b = [int(c * 255) for c in colorsys.hsv_to_rgb(h, 1.0, 1.0)]` converts our HSV colours to RGB. Because the `hsv_to_rgb` function both accepts and returns colour values between 0 and 1, and we need numbers between 0 and 255, we're using a list comprehension to multiply the RGB values we get back by 255 and also convert them back into integers (where it says `int(c * 255)`). List comprehensions are really handy ways to run the same operation on several things at once and take the form `[function(x) for x in y]`. They return a list. We're also using list unpacking (`r, g, b =`) to assign the resulting list of three values straight to three variables `r`, `g`, and `b`.

Finally, we're just using `set_all` to set the RGB values on all of the pixels, and the `show` function to update the pixels and show the colour, with a small delay of 0.02 seconds so that the rainbow cycles once every two seconds (since we have about 100 values).



The 14-segment displays

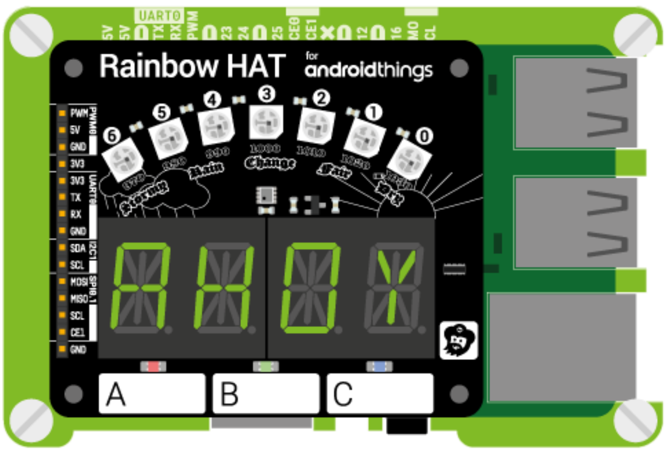
Because we've used 14-segment displays on Rainbow HAT rather than the more commonly-used 7-segment displays, we can display both numbers and upper and/or lower case text on them. There's also four decimal points that come in handy when displaying floating point numbers.

All of the display functions are collected together into the `display` module, and be accessed under `rh.display`. The two functions we'll be looking at here are `print_str` and `print_float`, which print strings and floating point numbers respectively, as you may have guessed!

We're going to print the string `AHOY` to the displays. Similar to the rainbow pixels, the displays have a `show` method, so after you've used `print_str` you also have call `show` to update the displays with your string.

Type the following:

```
rh.display.print_str('AHOY')

rh.display.show()
```



As well as being able to display text and numbers, there's support for most of the commonly-used special characters like @ and such.

Now, we'll use the `print_float` function to display some floating point numbers, inside a `while` loop that adds 0.01 each time, in effect creating a counter.

Type the following, pressing control-c to break out:

```
i = 0.0


while i < 999.9:

    rh.display.clear()

    rh.display.print_float(i)

    rh.display.show()

    i += 0.01
```

This starts at 0.0, clears the display, then uses the `print_float` method and `show` to display the current value of `i`, and finally increments `i` by 0.01.

You can also control the four decimal points, which is especially handy if you want to do something like make a clock. The `set_decimal` function takes two arguments - the position of the decimal point you want to toggle on, and a `True` or `False` value depending on whether you want to turn it on or off. The decimal points are numbered 0-3 from left to right.

Type the following to toggle on the first decimal point:

```
rh.display.clear()

rh.display.set_decimal(0, True)

rh.display.show()
```

Touch pads and LEDs

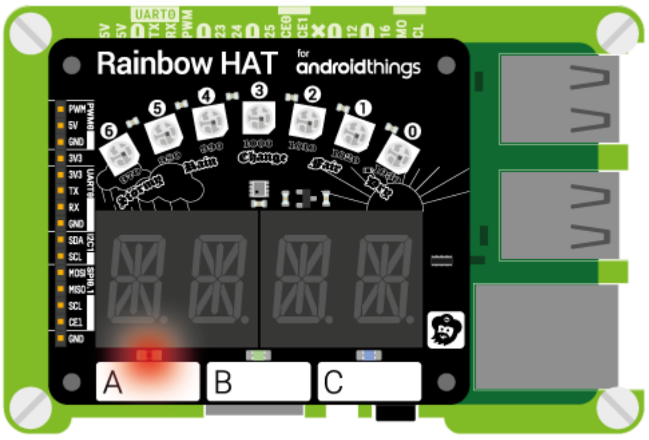
Rainbow HAT has three capacitive touch pads, marked A, B, and C, each with a coloured LED above. The touch pads can be used with decorator functions to link a touch and/or release event to a function. Decorators in Python are added directly before your function definition, allowing the function to be called automatically when the decorator is triggered, and can pass data into the function as well.

The LEDs associated with the touch pads can be controlled independently. Here, we'll link the LED above touch pad A to the touch pad, so that when the touch pad is pressed the LED lights up.

Type the following:

```
@rh.touch.A.press()

def touch_a(channel):

    print('Button A pressed')

    rh.lights.rgb(1, 0, 0)



@rh.touch.A.release()

def release_a(channel):

    print('Button A released')

    rh.lights.rgb(0, 0, 0)
```

You'll see that the touch pads - A, B, and C - are referred to in the decorator line as `@rh.touch.A`, `@rh.touch.B`, and `@rh.touch.C`, and that they have both `press()` and `release()` methods. Our two functions - `touch_a` and `release_a` are triggered by the generator when the pad is touched or released, and we light the LED with `rh.lights.rgb(1, 0, 0)`.



To control the LEDs above the pads, you pass either a 1 (to switch on) or a 0 (to switch off) to the `rh.lights.rgb`function. The three numbers passed correspond to each of the three LEDs in turn, meaning that you can toggle several on or off at the same time.

Type the following to turn all three LEDs on:

```
rh.lights.rgb(1, 1, 1)
```

And type the following to switch them off again:

```
rh.lights.rgb(0, 0, 0)
```

Temperature and pressure

The BMP280 sensor on Rainbow HAT measures both temperature and pressure. It's really simple to read the current temperature and pressure. Both are under the `rh.weather` module, so temperature can be read with `rh.weather.temperature()` and pressure with `rh.weather.pressure()`.

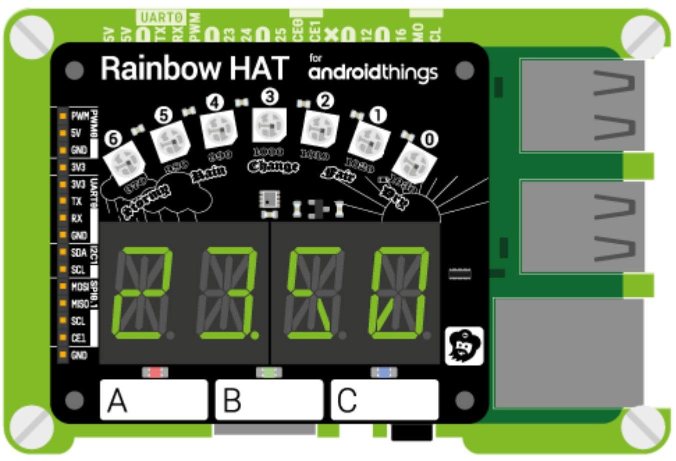
Type the following, pressing control-c to break out:

```
while True:

    t = rh.weather.temperature()

    p = rh.weather.pressure()

    print(t, p)

    time.sleep(0.5)
```

That should print the temperature in degrees Celsius and the pressure in Pascals (divide by 100 to get the more commonly-used hectoPascals) every half second.

You could easily modify that code to display the temperature on the 14-segment displays, using the `print_float`method, as follows:

```
while True:

    t = rh.weather.temperature()

    rh.display.clear()

    rh.display.print_float(t)
    rh.display.show()
    time.sleep(0.5)
```



Piezo buzzer

The piezo buzzer allows you to play notes of specific frequencies, or midi note numbers. Let's begin by playing a note of a specific frequency, middle C which has a frequency of 261Hz. The buzzer methods are under the buzzer module, `rh.buzzer`, and the `note` method takes a frequency in Hz and a note length in seconds.

Type the following:

```
rh.buzzer.note(261, 1)
```

That should play middle C for 1 second. Next, we'll do that same but with the `midi_note` function that allows you to use midi note numbers. Middle C is midi note 60, so we'll play that now.

Type the following:

```
rh.buzzer.midi_note(60, 1)
```

To play a tune, you could make a list of the notes, and then loop through them, with a short pause after each one.

Type the following:

```
song = [68, 68, 68, 69, 70, 70, 69, 70, 71, 72]

for note in song:
    rh.buzzer.midi_note(note, 0.5)
    time.sleep(1)
```

It's worth noting that if you want a half second note and then a half second pause, you'll add your note length to the length of your pause, e.g. 0.5 + 0.5 = 1 in the example above.

Also, the notes we've used are all of the same length and ideally you'd want to vary the note length to make the tune sound correct.

Did you guess what song that was?

Project ideas

The variety of different inputs and outputs on Rainbow HAT makes it ideal for all sorts of projects, combining the inputs and outputs in interesting ways and applying what you've learned above.

Here are some ideas we've had:

Weather station

Use the temperature and pressure sensor, and display the values on the 14-segment displays. You could even use the rainbow LEDs to display a bar chart of the temperature or pressure with the colour indicating cold or warm temperatures, or low or high pressure. Why not make it internet-connected and tweet the values, or log the values to a Google Sheet?

Tiny, three-note piano

There are plenty of tunes you can play with three notes! Here’s one to get you started - Mary Had a Little Lamb. The piezo buzzer uses PWM (pulse width modulation) to generate different frequencies of sound, and it’s easy to map these frequencies to notes. You could even use chording to get up to eight notes!

Reaction time game

Use the buttons to make a reaction time game for up to three players. Press a button to start, and when the rainbow lights change from red to green the first player to hit their button is the winner. Display the winning time on the 14-segment displays.

Stopwatch or countdown timer

Use the buttons as controls for starting or stopping a stopwatch, and display the time on the 14-segment displays. You could use the rainbow LEDs to display the seconds ticking. Or use the buttons to set the seconds and minutes on a countdown timer, displaying the countdown on the displays, and then buzz the buzzer when the time is up!

Mood light

Set your mood with the rainbow LEDs, using the 3 buttons to select and adjust each colour channel, fade through the hue, or switch between lighting effects.