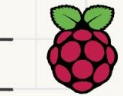
Making Gadgets with Python

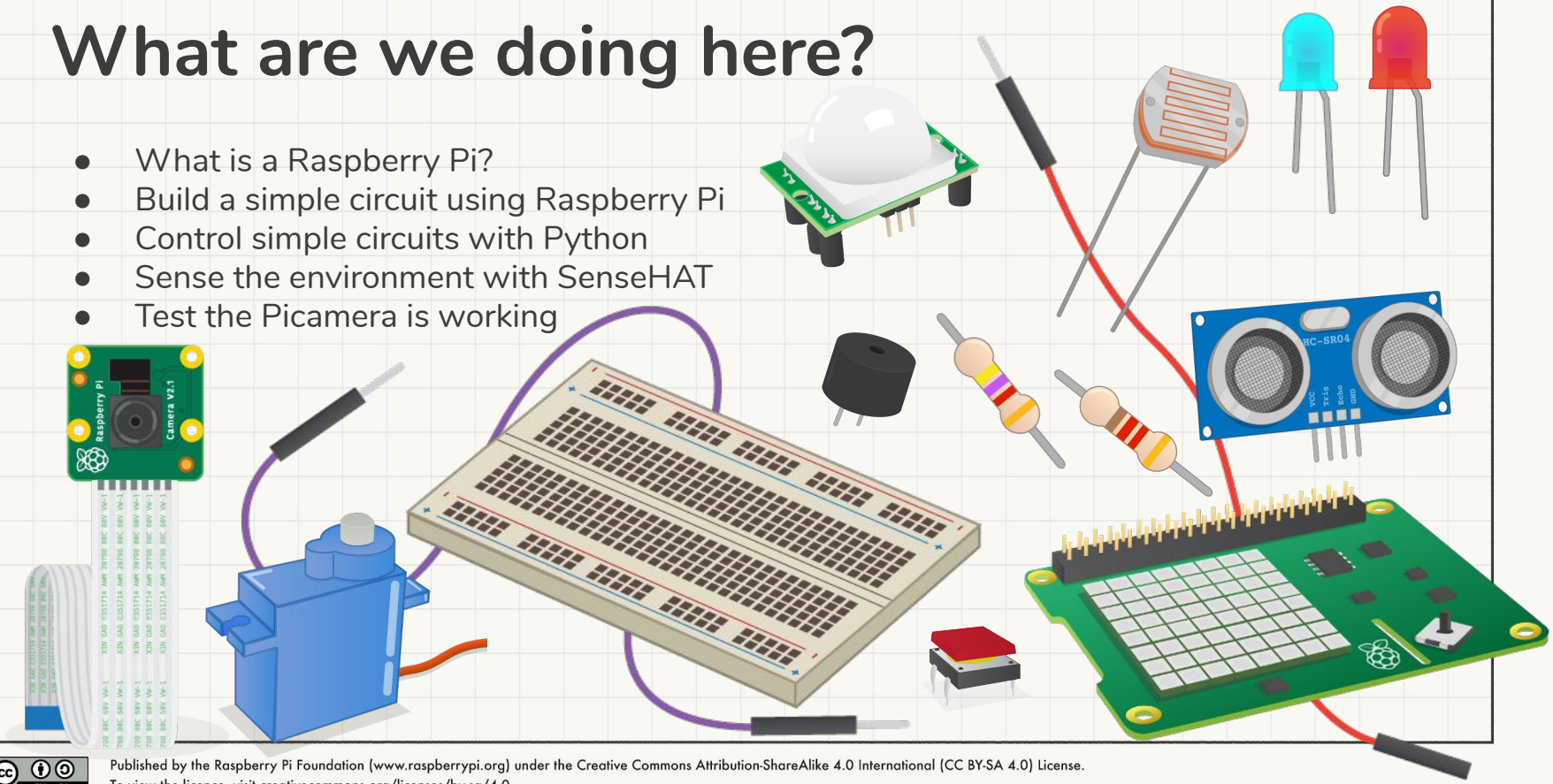
“To master a new technology, you have to play with it.”

- Jordan Peterson

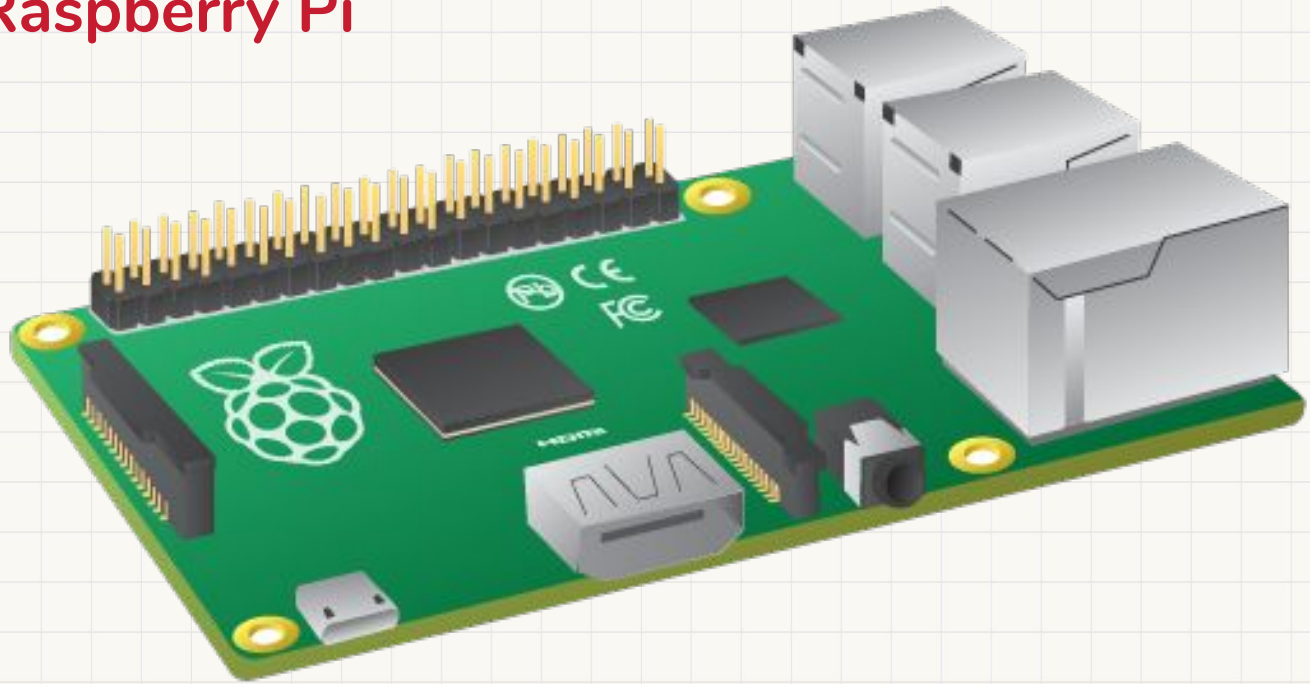


What are we doing here?

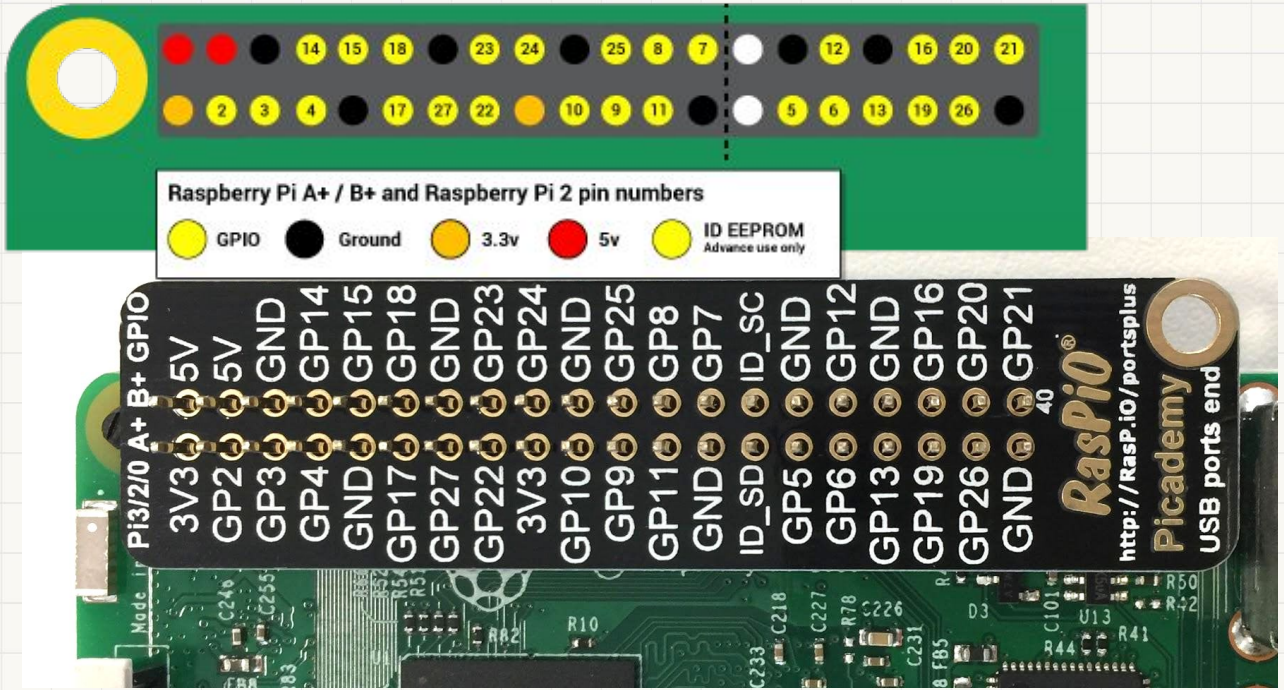
- What is a Raspberry Pi?
- Build a simple circuit using Raspberry Pi
- Control simple circuits with Python
- Sense the environment with SenseHAT
- Test the Picamera is working



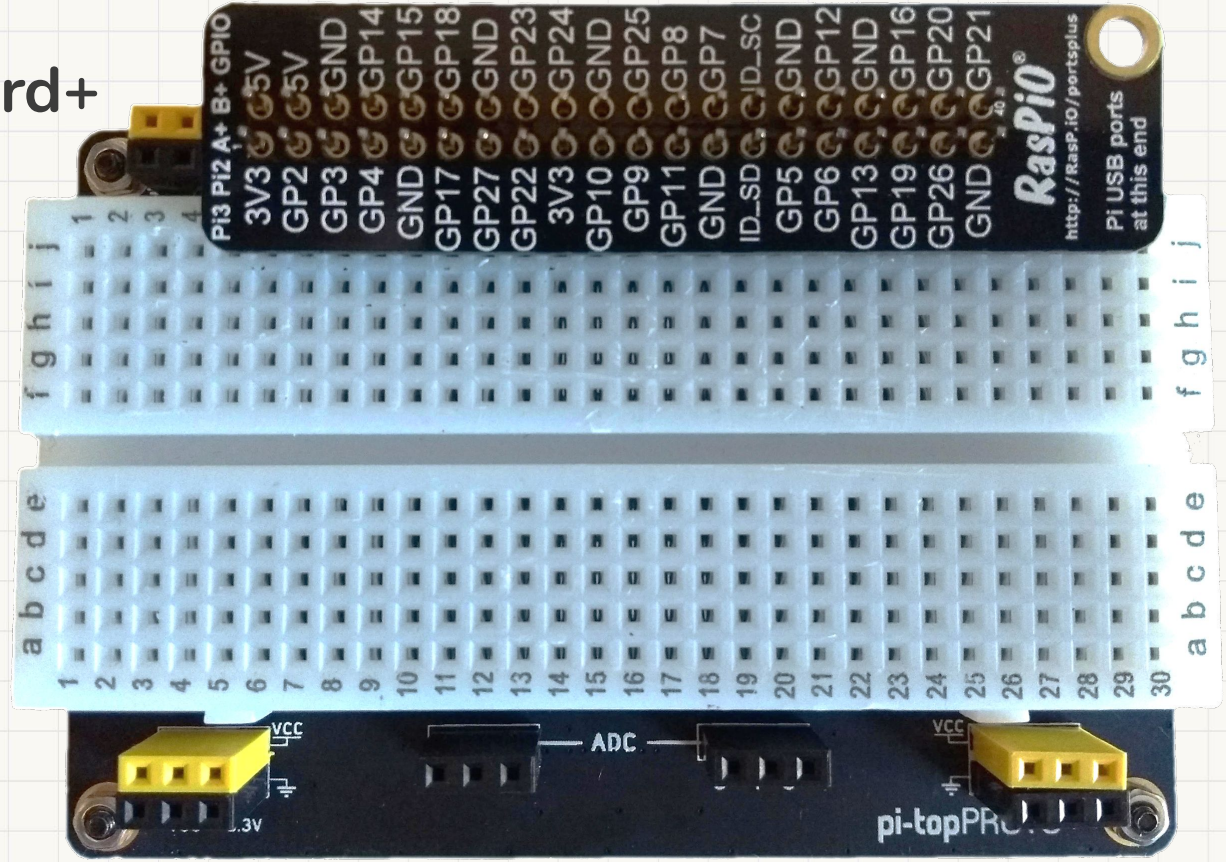
The Raspberry Pi



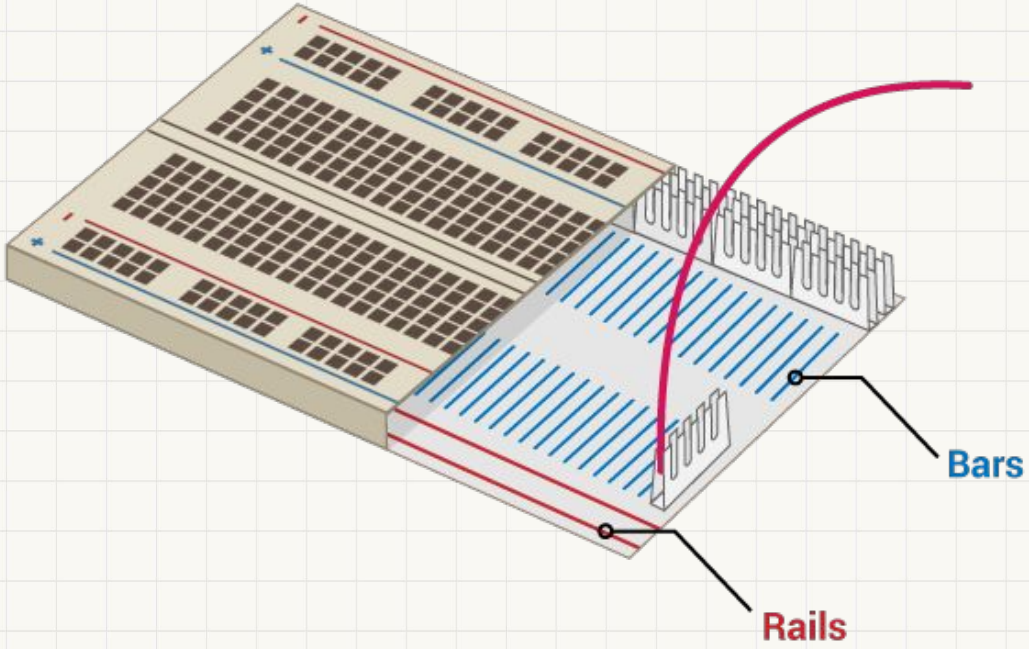
Physical computing



Protoboard+

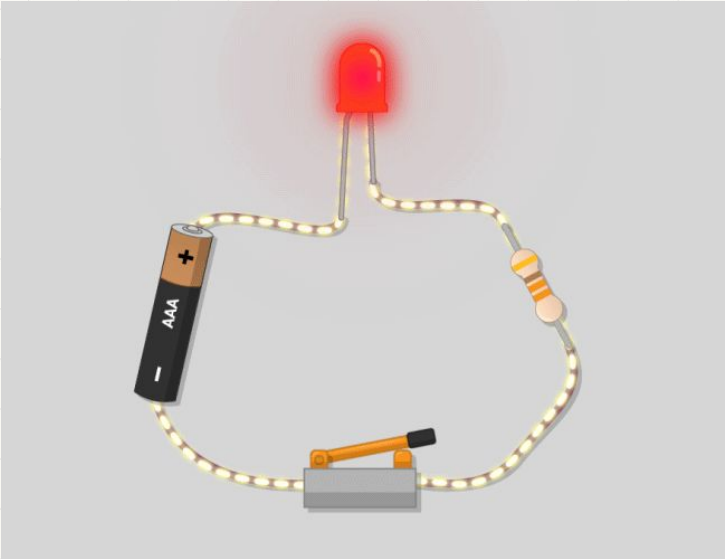


Breadboards



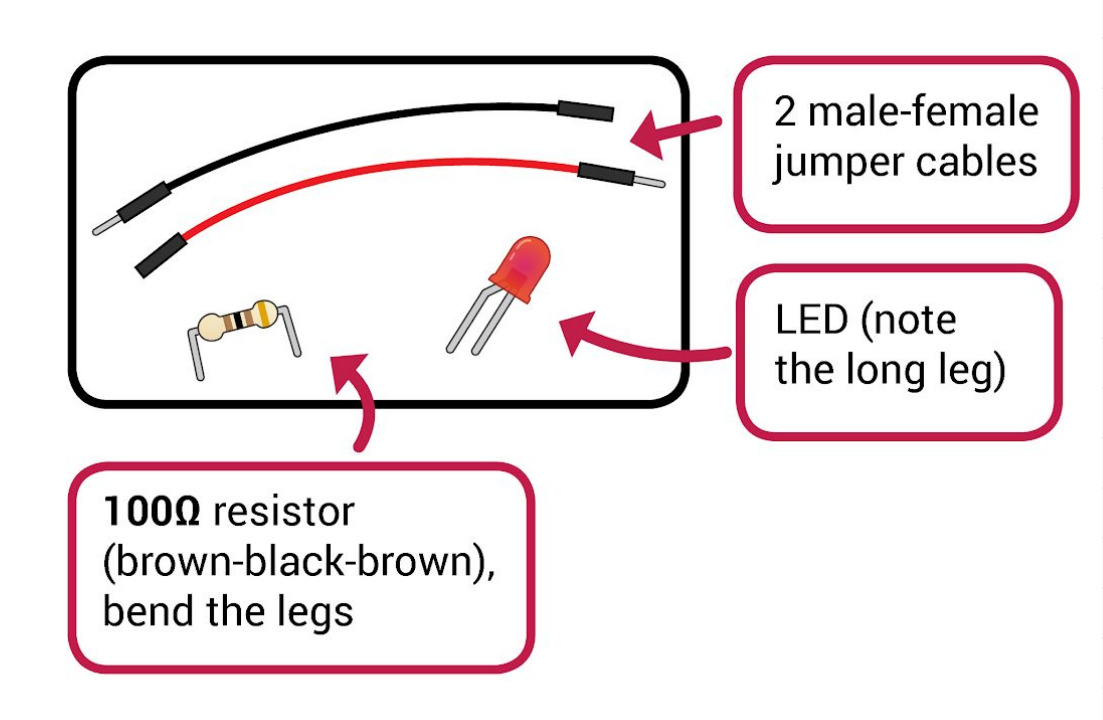
Circuits

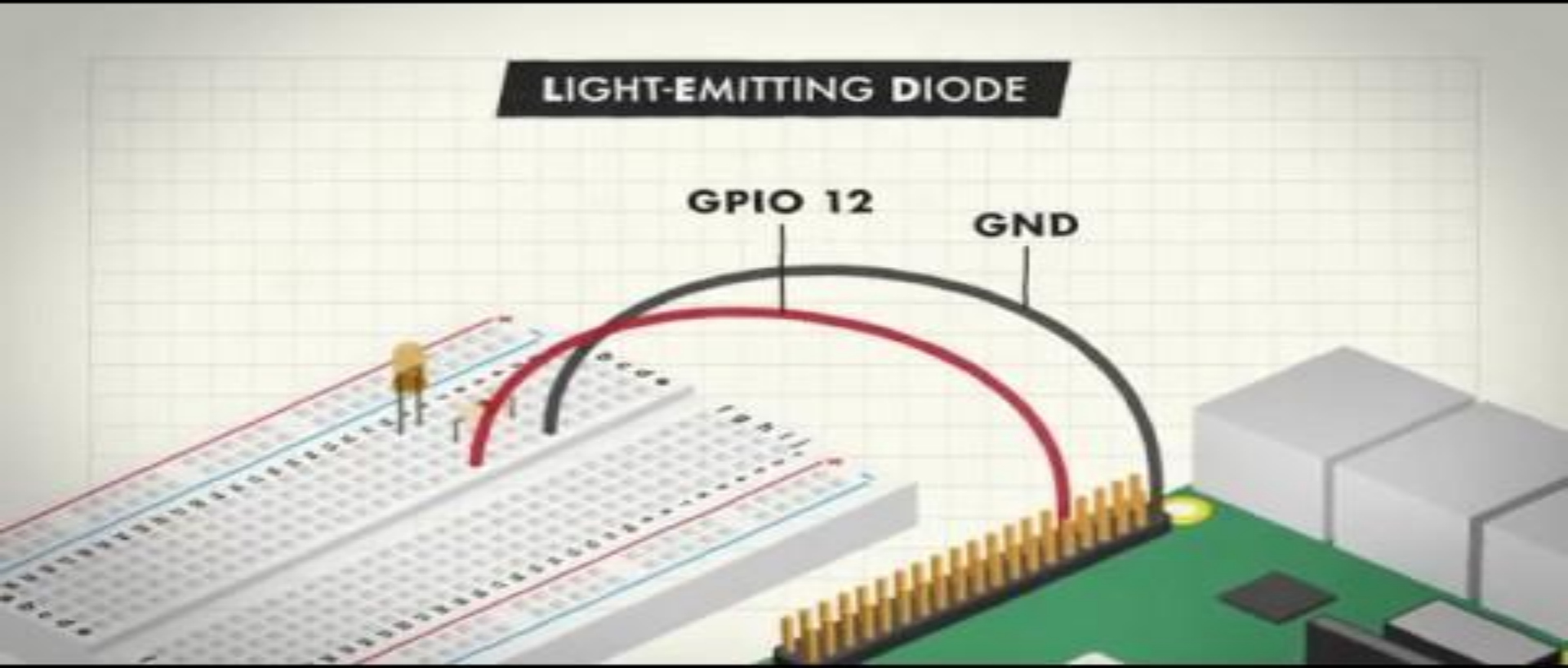
Your Raspberry Pi can act as the power supply for simple circuits. We can use this to test our hardware is working.



Connect an LED

You will need to collect these components:



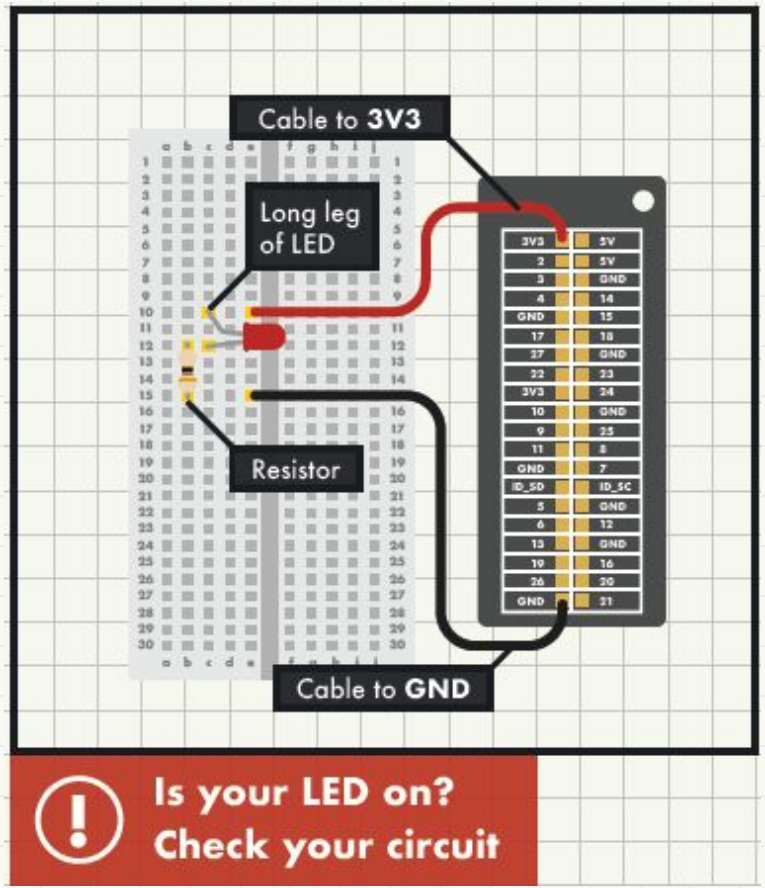

LIGHT-EMITTING DIODE
GPIO 12
GND

Test an LED

Here, we wire our LED directly to a 3.3 volt pin, which is **always on**.

Follow this diagram **exactly** to light your LED!

Once your LED is lit, open the Mu Editor from your launch bar:



Anatomy of a Python Script

```
from gpiozero import Button, LED
from time import sleep
```

At the beginning of our script, we **import** all the libraries we will need, listing all the components we will use, split by commas.

```
led = LED(12)
btn = Button(10)

def alarm():
    led.blink()
    print('Intruder!!')
    sleep(5)
    led.off()
```

Next, we **set up** our components with pin numbers & define our functions.

This function is called `alarm()`. This is the list of instructions we want the Pi to complete when we push the button.

```
while True:
    btn.wait_for_press()
    print('button pressed')
    alarm()
```

Finally, we write the part of the code that actually runs, called the **execution**. You can call on any previously defined functions or components here. This is the part that makes the machine work.

Control an LED

Now, move the positive connection from the 3V pin to **any green GPIO pin** on your header. Here, **we have used pin 12**, so it is the number we put in the code when we tell the Pi which pin to use:

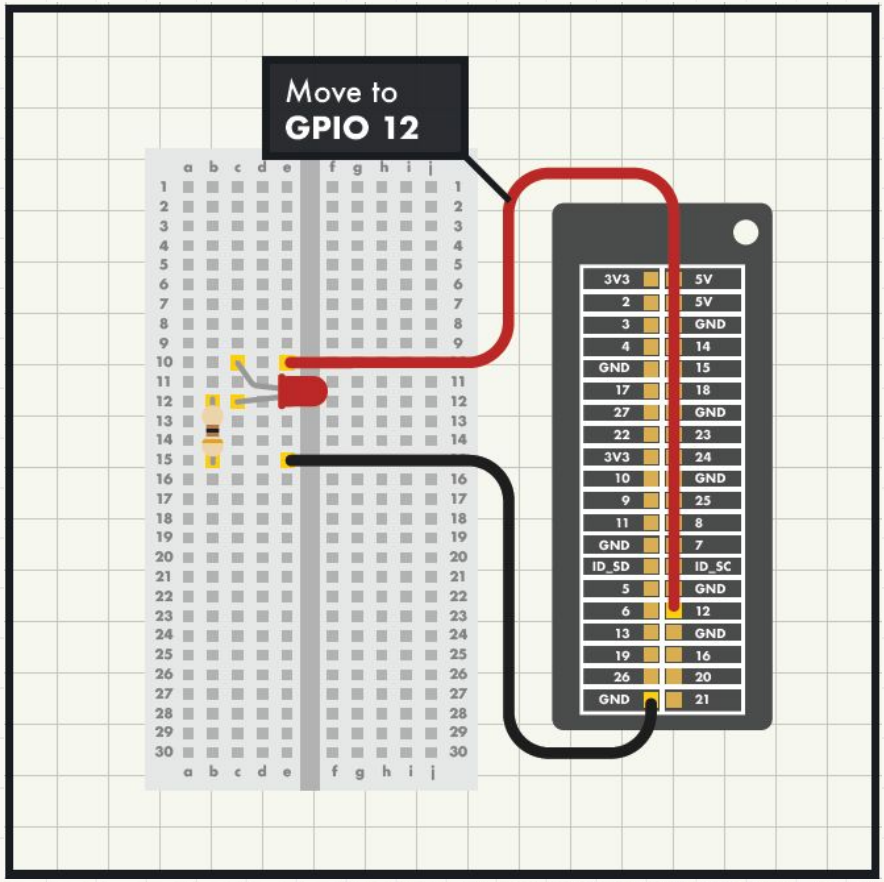
Pin # 

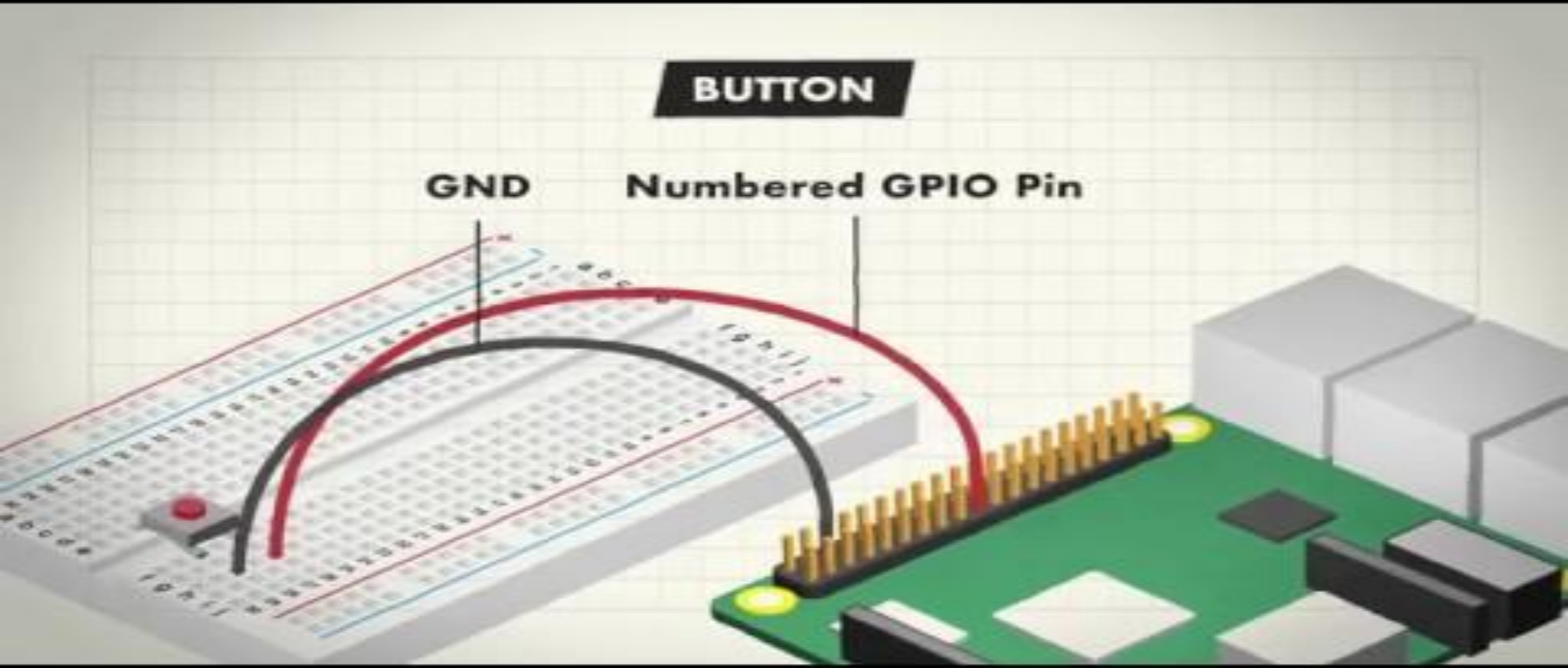
`from gpiozero import LED` `from time import sleep`	**Import**
`red = LED(12)`	**Setup**

`red.on()`
`sleep(3)`
`red.off()`
	Execute

OR	
`red.blink()`	**Execute**

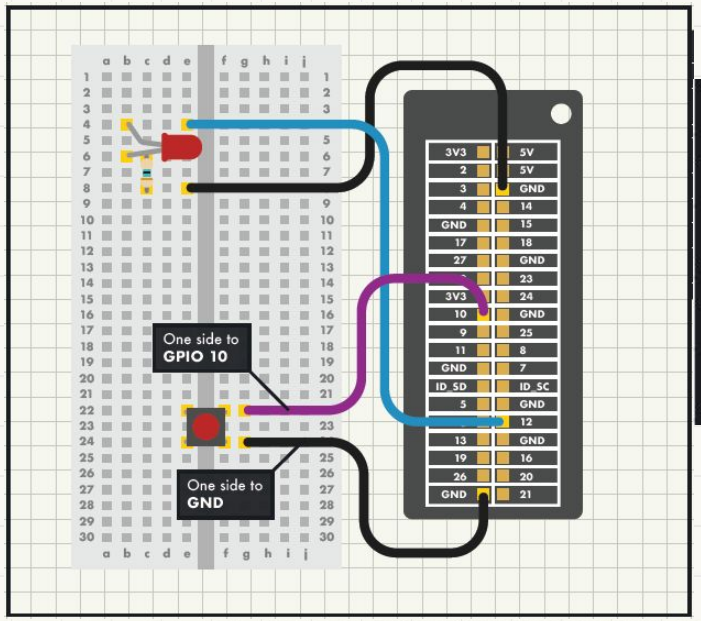
Once you're done, save your code as **`test1.py`** and click the green **Run** triangle to execute it!




BUTTON
GND
Numbered GPIO Pin

Control an LED with a button

Don’t write a whole new script! Just add to and change **`test1.py`** so it looks exactly like the following:




```
from gpiozero import LED, Button
```
Import

```
red = LED(12)
btn = Button(10)
```
Setup

```
btn.when_pressed = red.on
btn.when_released = red.off
```
Execute

You can also copy+paste your code into a new script - don’t work harder, work smarter!

Control an LED with a button

Your scripts can have different **executions** of the same components.

The execution section determines **what happens** when your program runs.

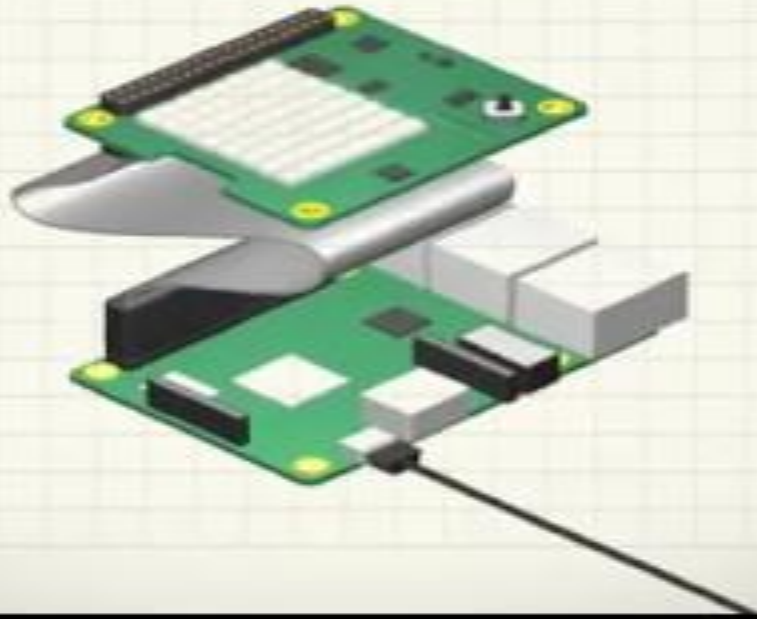
Notice that the **import** and **setup** sections are exactly the same in these two scripts; only the last part is different.

The **import** and **setup** sections organise and ready your components and code to be **executed** on Run.

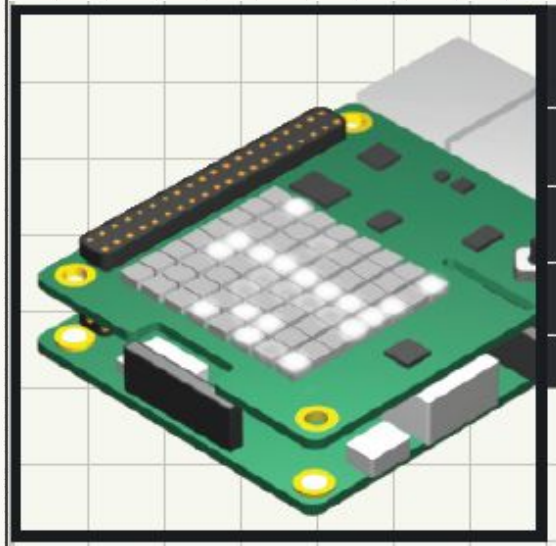
`from gpiozero import LED, Button`	**Import**
`red = LED(12)` `btn = Button(10)`	**Setup**
`btn.when_pressed = red.toggle`	**Execute**

`from gpiozero import LED, Button`	**Import**
`red = LED(12)` `btn = Button(10)`	**Setup**
`btn.when_pressed = red.blink` `btn.when_released = red.off`	**Execute**

SENSE HAT



SenseHAT pixels



Show message

`from sense_hat import SenseHat`	**Import**
`sense = SenseHat()`	**Setup**
`sense.show_message("Hello World!")`	**Execute**

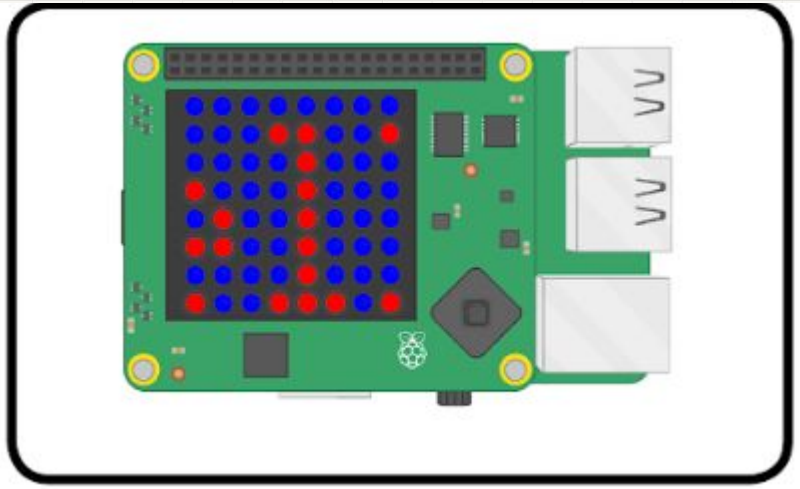
Colour mixing

Colours can be created by mixing different levels of Red, Green and Blue.
We can set the LED array to display any colours we want using this system, allowing us to make around 16 million different hues!

You can easily find lists of these values on the internet - just look for ‘RGB colours’.

SenseHAT pixels

By changing the `text_colour` and `back_colour` parameters in our code, we can make the words and background whatever colour we like:

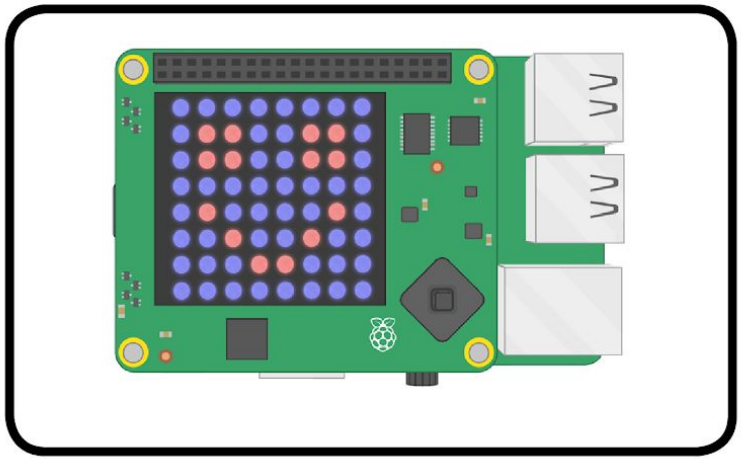


Remember: You can't put in a colour value more than 255 or it won't work!

```
sense.show_message("Hello World!", text_colour=(255,0,0), back_colour=(0,0,255))
```

SenseHAT pixels

We can also use the SenseHAT to draw images in our 8x8 grid. Simply setup the colours you want to use, define your image and ‘call’ it like this:



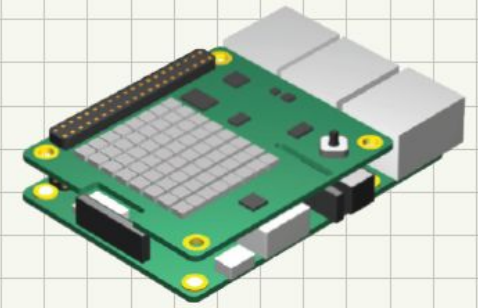
```
r = (255, 0 ,0)
b = (0, 0, 255)

smile = [b,b,b,b,b,b,b,b,
         b,r,r,b,b,r,r,b,
         b,r,r,b,b,r,r,b,
         b,b,b,b,b,b,b,b,
         b,r,b,b,b,b,r,b,
         b,b,r,b,b,r,b,b,
         b,b,b,r,r,b,b,b,
         b,b,b,b,b,b,b,b]

sense.set_pixels(smile)
```

SenseHAT sensors

The SenseHAT can also detect the environment around it, and is used on the ISS! Open a new script in Mu, then save and Run the following:



Display pressure, temperature and humidity

`from sense_hat import SenseHat`	**Import**

`sense = SenseHat()`
 	
`pressure = sense.get_pressure()`
`temp = sense.get_temperature()`
`humidity = sense.get_humidity()`
	Setup

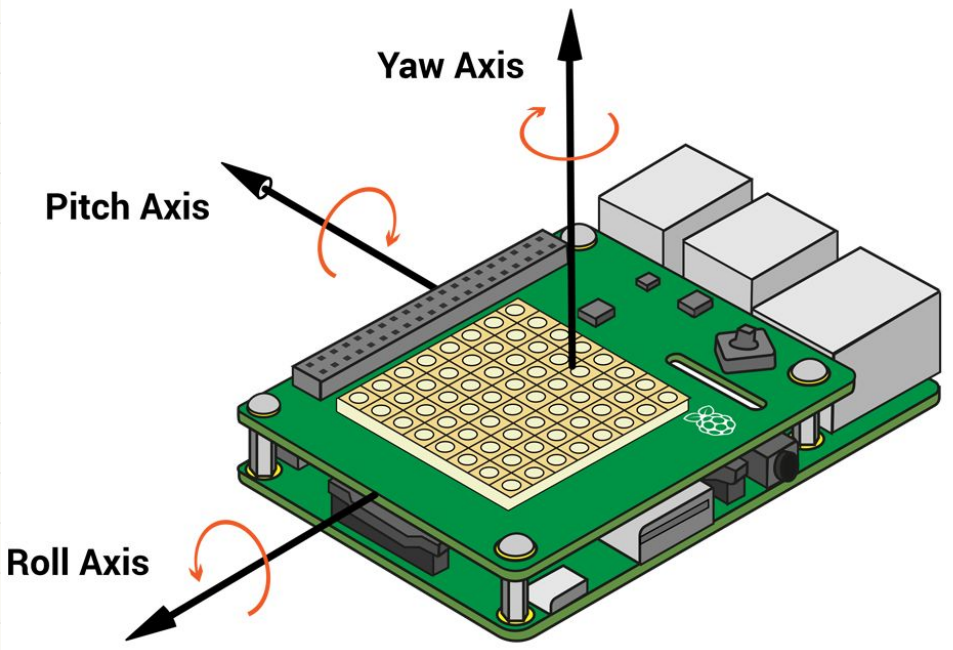
`print(pressure)`
`print(temp)`
`print(humidity)`
	Execute

```
963.994056234
54.6666666667
71.8333333333
>>>
```

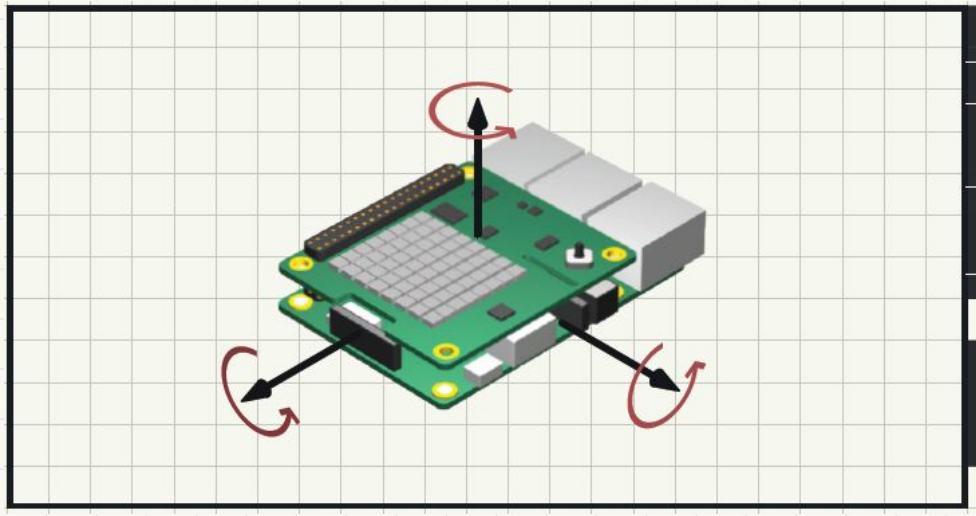
Detecting pitch, roll, and yaw

The SenseHAT can also detect its' orientation at any time.

It can also detect if it receives a shock or is hit or dropped.



SenseHAT sensors



Display orientation

`from sense_hat import SenseHat`	**Import**

`sense = SenseHat()`
 	
`orientation = sense.get_orientation()`
	Setup

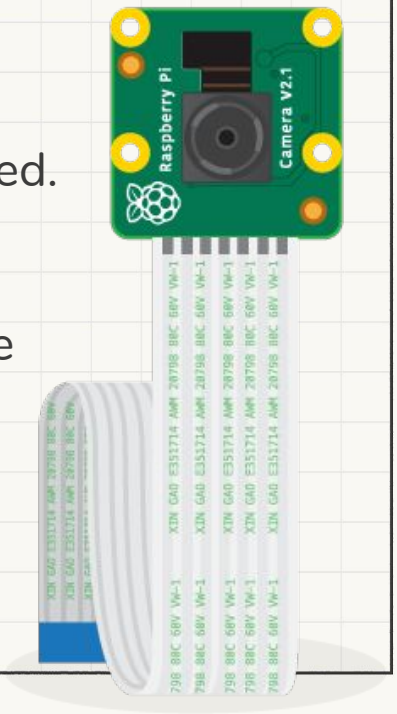
`print(orientation["pitch"])`
`print(orientation["yaw"])`
`print(orientation["roll"])`
	Execute

```
345.930712043
105.245539221
299.287767481
>>>
```

PiCamera

Your camera is connected to the Raspberry Pi by a white ribbon cable: take it out and place it on the table.

Please be gentle with it, as the cables can snap if mistreated.



Test Your Camera:

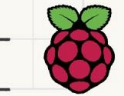
1. Open the terminal from your launch bar by clicking the icon: 
2. Into the terminal window type: **`raspistill -k`**
3. Press Enter to see the camera's view
4. Press `Ctrl+C` to close the preview window

Gadget 'Recipes'

Project	Components	Function
Nightlight	LED + PIR/UDS	Detects motion/Proximity turns on light
Fall Alarm	SenseHAT (accel) + buzzer	Detects shock/fall sends alert/web based message
Occupancy Light	LED + PIR	Detects motion/Proximity turns on light
Medication Timer	SenseHAT/ LEDs+Buzzer	Displays reminders at certain times
Proximity Sensor	LED/Buzzer + UDS	Detects close proximity, sounds buzzer
Security Camera	Picamera + PIR/UDS	Detects motion, captures footage
Photobooth	Picamera + Button/PIR/UDS	Take selfies with a button
Out of Bed Alert	Button/Tactile switch + LED/Buzzer	Switch on the floor to tell staff when someone gets out of bed
Help Alarm/Panic Button	Button + buzzer/internet post	Press button to sound alarm or send a message
Environment Sensor	SenseHAT	Sense the temp/humidity in the room
Occupancy Counter	UDS/PIR	Count how many people pass the Pi
Bop-it Game	SenseHAT	Create a reaction game using sensehat accelerometer and sensors
Time Lapse Camera	Picamera	Take timed photos over a period
Reaction Testing Game	Buttons	Create a reaction game with physical buttons
Burglar Alarm	PIR + Buzzer + Picamera	Create a motion detecting burglar alarm
Security Light	LED+PIR	Motion activated light
Timed Light	LED	Light set to come on at certain times.

Pack away!

All components back in the kits, laptops powered off.



Raspberry Pi