

Raspberry Pi Sense Hat

Inhaltsverzeichnis

LED-Matrix.....	3
set_pixels.....	3
get_pixels.....	3
set_pixel.....	3
clear.....	4
show_message.....	5
show_letter.....	5
low_light.....	5
gamma.....	5
gamma_reset.....	5
Umgebungssensoren.....	6
get_temperature.....	6
get_humidity.....	6
get_temperature_from_humidity.....	6
get_pressure.....	6
get_temperature_from_pressure.....	6
Trägheitssensoren.....	7
set_imu_config.....	7
get_orientation_radians.....	7
get_orientation_degrees.....	7
get_orientation.....	7
get_compass.....	7
get_compass_raw.....	8
get_gyroscope.....	8
get_gyroscope_raw.....	8
get_accelerometer.....	8
get_accelerometer_raw.....	9
Joystick.....	9
wait_for_event.....	9
get_events.....	10
direction_up,.....	10
direction_left,.....	10
direction_right,.....	10
direction_down,.....	10
direction_middle,.....	10
direction_any.....	10

Als Ausgangspunkt habe ich die API-Referenz von [pythonhosted.org](https://pythonhosted.org/sense-hat/api/) genutzt.

Quelle: <https://pythonhosted.org/sense-hat/api/>

Für die Nutzung des Sense HAT muss die zugehörige Software installiert sein.

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

Um die angezeigten Beispiele zu nutzen, muss die Sense HAT Bibliothek vorher über folgende Befehle initialisiert werden bzw. den hier aufgezeigten Programmbeispielen vorangestellt werden:

Sense Hat Bibliothek importieren / initialisieren

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

LED-Matrix

set_pixels

Aktualisiert die gesamte LED-Matrix basierend auf einer Liste mit 64 Pixelwerten mittels R, G, B Werte.

```
sense.set_pixels(led_matrix)
```

```
X = [255, 0, 0]
O = [255, 255, 255]

led_matrix = [
    0, 0, 0, X, X, 0, 0, 0,
    0, 0, X, 0, 0, X, 0, 0,
    0, 0, 0, 0, 0, X, 0, 0,
    0, 0, 0, 0, X, 0, 0, 0,
    0, 0, 0, X, 0, 0, 0, 0,
    0, 0, 0, X, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, X, 0, 0, 0, 0
]
```

```
sense.set_pixels(question_mark)
```

get_pixels

Liest die Farbwerte **aller** 64 LED-Matrix als R, G, B Werte in eine Liste ein.

```
farbliste = sense.get_pixels()
```

Die Werte werden in der Liste wie folgt gespeichert:

```
[[0, 0, 0], [0, 0, 0], ... , [0, 0, 0], [0, 0, 0]]
```

set_pixel

Setzt ein einzelnes LED-Matrixpunkt an der angegebenen X-Y-Koordinate mit der angegebene Farbe.

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

x = X-Koordinate 0 - 7

y = Y-Koordinate 0 - 7

r = Rotwert 0 - 255

g = Grünwert 0 - 255

b = Blauwert 0 - 255

```
sense.set_pixel(7, 7, 255, 255, 255)
```

```
sense.set_pixel(0, 0, 0, 255, 0)
```

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

```
green = (0, 255, 0)
```

```
blue = (0, 0, 255)
```

```
sense.set_pixel(0, 0, red)
```

```
sense.set_pixel(0, 0, green)
```

```
sense.set_pixel(0, 0, blue)
```

get_pixel

Gibt den Farbwert als Liste (r, g, b) für den Bildpunkt der LED-Matrix für die angegebenen Koordinaten zurück.

```
sense.get_pixel(x, y)
```

x = X-Koordinate 0 – 7
y = Y-Koordinate 0 – 7

```
farbe_pixel = sense.get_pixel(0, 0)
```

set_rotation

Die LED-Matrix Ausgabe wird gedreht. Der Winkel kann mit 0, 90, 180 & 270 angegeben werden.

```
sense.set_rotation(90)
```

alternativ:

```
sense.rotation = 90
```

flip_h

Spiegelt die LED-Matrix horizontal. Zusätzlich wird eine Liste zurückgegeben mit allen 64 R, G, B Werten der einzelnen Matrix-Punkten. Die ist entspricht wie dem Befehl get_pixels.

```
sense.flip_h()
```

flip_v

Spiegelt die LED-Matrix vertikal. Zusätzlich wird eine Liste zurückgegeben mit allen 64 R, G, B Werten der einzelnen Matrix-Punkten. Die ist entspricht wie dem Befehl get_pixels.

```
sense.flip_v()
```

load_image

Lädt eine Bilddatei, konvertiert sie in das RGB-Format und zeigt sie auf der LED-Matrix an. Das Bild muss 8 x 8 Pixel groß sein. Zusätzlich wird eine Liste zurückgegeben mit allen 64 R, G, B Werten der einzelnen Matrix-Punkten. Die ist entspricht wie dem Befehl get_pixels.

```
sense.load_image("bildname.png")
```

clear

Löscht die LED-Matrix mit er angegebenen Farbe. Wird kein Wert angegeben werden die LEDs abgeschalten.

```
sense.clear()
```

Ohne Argumente löscht die LED-Matrix

```
red = (255, 0, 0)  
sense.clear(red)
```

Übergabe mittels Tupel

```
sense.clear(255, 255, 255)
```

Übergabe mittels r, g, b Werte

show_message

Zeigt einen Scrolltext von rechts nach links über die LED-Matrix mit der angegebenen Geschwindigkeit in der angegebenen Text- und Hintergrundfarbe an. Werden keine Farbwerte angegeben, wird der Text auf weiß angezeigt ohne Hintergrundfarbe. Ohne die Angabe der Scrollgeschwindigkeit beträgt der Standardwert 0,1.

```
sense.show_message("Test Text!", text_colour=[255, 0, 0], back_colour=[255, 255, 255])
```

```
sense.show_message("Test Text!", text_colour=[255, 0, 0])
sense.show_message("Test Text!", back_colour=[255, 0, 0])
sense.show_message("Test Text!", scroll_speed=0.1)
```

show_letter

Zeigt einen einzelnen Buchstaben in Form eines Strings mit der angegebenen Text- und Hintergrundfarbe an. Werden keine Farbwerte angegeben, wird der Text auf weiß angezeigt ohne Hintergrundfarbe.

```
sense.show_letter("P", text_colour=[255, 0, 0], back_colour=[255, 255, 255])
```

low_light

Ermöglicht das Abdunkeln der LED-Matrix.

```
sense.low_light = True
sense.low_light = False
```

Dunkelt die LED-Matrix ab.
Deaktiviert den low_light-Modus.

gamma

Mit gamma kann eine Nachschlagetabelle für die letzten 5 verwendeten Farbbits angegeben werden. Die Nachschlagetabelle ist eine Liste mit 32 Zahlen, die zwischen 0 und 31 liegen müssen. Der Wert der eingehenden 5-Bit-Farbe wird zum Indizieren der Nachschlagetabelle verwendet, und der an dieser Position gefundene Wert wird dann in die LEDs geschrieben.

```
sense.gamma = [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,3,3,3,4,4,5,5,6,6,7,7,8,8,9,10,10]
```

gamma benötigt einiges an Testaufwand, um geeignete Werte zu finden. Daher ist der Befehl low_light für die meisten Benutzer ausreichend.

gamma_reset

Setzt die Gamma-Lookup-Tabelle auf die Standardwerte zurück.

```
sense.gamma_reset()
```

Der Standardwert der Gammetabelle lautet:

```
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]
```

```
sense.gamma = [0] * 32
```

Schaltet die LED-Matrix aus, da die Gamma-Lookup-Tabelle für alle 32 Werte auf 0 gesetzt wird. sense.gamma_reset() würde die LED-Matrix wieder einschalten.

Umgebungssensoren

get_temperature

Ermittelt die Temperatur in Grad Celsius vom Temperatursensor.

```
print = sense.get_temperature()  
print("Temperature: %s C" % temp)
```

alternativ

```
print(sense.temp)  
print(sense.temperature)
```

get_humidity

Ermittelt die relative Luftfeuchte in Prozent gemessen durch den Luftfeuchtesensor.

```
luftfeuchte = sense.get_humidity()  
print("Humidity: %s %%rH" % luftfeuchte)
```

alternativ

```
print(sense.humidity)
```

get_temperature_from_humidity

Ermittelt die Temperatur in Grad Celsius des Temperatursensor vom Luftfeuchtesensor.

```
print(sense.get_temperature_from_humidity())
```

alternativ

```
temp = sense.get_temperature_from_humidity()  
print("Temperature: %s C" % temp)
```

get_pressure

Ermittelt den aktuellen Luftdruck vom Luftdrucksensor in Millibar.

```
pressure = sense.get_pressure()  
print("Pressure: %s Millibars" % pressure)
```

alternativ

```
print(sense.pressure)
```

get_temperature_from_pressure

Ermittelt die Temperatur in Grad Celsius des Temperatursensor vom Luftdrucksensor.

```
print(sense.get_temperature_from_pressure())
```

alternativ

```
temp = sense.get_temperature_from_pressure()  
print("Temperature: %s C" % temp)
```

Trägheitssensoren

Der IMU-Sensor (Inertial Measurement Unit | Messeinheit von Trägheitssensoren) ist eine Kombination aus drei Sensoren mit jeweils einer x-, y- und z-Achse. Die 3 verschiedenen Sensoren können einzeln oder in Kombination verwendet werden.

set_imu_config

Aktiviert (True) oder deaktiviert (False) die Sensoren des Gyroskops, Beschleunigungssensors und / oder des Magnetometers (Kompass).

```
sense.set_imu_config(compass_enabled = True, gyro_enabled = False, accel_enabled = True)
```

alternativ

```
sense.set_imu_config(True, False, True)
```

Es müssen immer alle drei Parameter angegeben werden.

get_orientation_radians

Ermittelt die aktuelle Ausrichtung in Bogenmaß unter Verwendung der drei Hauptachsen: Querachse (Nicken – pitch), Längsachse (Rollen - roll) und Vertikale Achse (Gieren – yaw).

```
print(sense.orientation_radians)
```

alternativ

```
ausrichtung_rad = sense.get_orientation_radians()
print("p: {pitch}, r: {roll}, y: {yaw}".format(**ausrichtung_rad))
```

get_orientation_degrees

Ermittelt die aktuelle Ausrichtung in Grad unter Verwendung der drei Hauptachsen: Querachse (Nicken – pitch), Längsachse (Rollen - roll) und Vertikale Achse (Gieren – yaw). Die Wörterbuchobjekte pitch, roll und yaw werden automatisch indiziert.

```
ausrichtung = sense.get_orientation_degrees()
print("p: {pitch}, r: {roll}, y: {yaw}".format(**ausrichtung))
```

get_orientation

Ruft den Befehl `get_orientation_degrees` auf (siehe oben).

```
print(sense.orientation)
```

alternativ

```
ausrichtung = sense.get_orientation()
print("p: {pitch}, r: {roll}, y: {yaw}".format(**ausrichtung))
```

get_compass

Ruft `set_imu_config` zum Deaktivieren des Gyroskopsensors sowie des Beschleunigungssensors und ermittelt die Nordausrichtung durch das Magnetometer in Grad.

```
print(sense.compass)
```

alternativ

```
norden = sense.get_compass()
```

```
print("Norden: %s" % norden)
```

get_compass_raw

Ermittelt die Rohdaten der x, y und z-Achse des Magnetometers. Die Wörterbuchobjekte x, y und z werden automatisch indiziert. Die Werte sind Fließkommazahlen und zeigen die magnetische Indizität in Microtesla (μT) an.

```
print(sense.compass_raw)
```

alternativ

```
raw = sense.get_compass_raw()
print("x: {x}, y: {y}, z: {z}".format(**raw))
```

get_gyroscope

Ruft `set_imu_config` zum Deaktivieren des Magnetometers sowie des Beschleunigungssensors und ermittelt die aktuelle Ausrichtung des Gyroskopsensors. Die Wörterbuchobjekte pitch, roll und yaw werden automatisch indiziert.

```
nur_gyro = sense.get_gyroscope()
print("p: {pitch}, r: {roll}, y: {yaw}".format(**nur_gyro))
```

alternativ

```
print(sense.gyro)
print(sense.gyroscope)
```

get_gyroscope_raw

Ermittelt die Rohdaten der x, y und z-Achse des Magnetometers. Die Wörterbuchobjekte x, y und z werden automatisch indiziert. Die Werte sind Fließkommazahlen und zeigen die Achsen in Bogenmaß pro Sekunde an.

```
raw = sense.get_gyroscope_raw()
print("x: {x}, y: {y}, z: {z}".format(**raw))
```

alternativ

```
print(sense.gyro_raw)
print(sense.gyroscope_raw)
```

get_accelerometer

Ruft `set_imu_config` zum Deaktivieren des Magnetometers sowie des Gyroskopsensors und ermittelt die aktuelle Ausrichtung des Beschleunigungssensors. Die Wörterbuchobjekte pitch, roll und yaw werden automatisch indiziert.

```
nur_beschleunigung = sense.get_accelerometer()
print("p: {pitch}, r: {roll}, y: {yaw}".format(**nur_beschleunigung))
```

alternativ

```
print(sense.accel)
print(sense.accelerometer)
```


get_accelerometer_raw

Ermittelt die Rohdaten der x, y und z-Achse des Beschleunigungssensors. Die Wörterbuchobjekte x, y und z werden automatisch indiziert. Die Werte sind Fließkommazahlen und zeigen die Beschleunigung in G an.

```
raw = sense.get_accelerometer_raw()
print("x: {x}, y: {y}, z: {z}".format(**raw))
```

alternativ

```
print(sense.accel_raw)
print(sense.accelerometer_raw)
```

Joystick

Eingabeereignis (InputEvent)

Die Abfrage des Joysticks gibt 3 Parameter als Tuple zurück.

timestamp	Der Zeitpunkt, zu dem das Ereignis aufgetreten ist. Die Ausgabe erfolgt im selben Format wie bei der time Funktion		
direction	Gibt die Bewegungsrichtung als String zurück	"up"	nach oben
		"down"	nach unten
		"left"	nach links
		"right"	nach rechts
		"middle"	nicht betätigt
action	Gibt ausgeführte Aktion als String zurück	"pressed"	gedrückt
		"released"	losgelassen
		"held"	haltend

Diese Informationen werden von mehreren Joystickabfragebefehlen bzw. -methoden entweder als Rückgabewert oder als Typ eines Parameters verwendet.

wait_for_event

Wartet mit der Ausführung eines Programmes bis ein Joystickeignis auftritt und gibt die entsprechenden Parameter zurück.

```
from time import sleep

sense = SenseHat()
event = sense.stick.wait_for_event()
print("The joystick was {} {}".format(event.action, event.direction))
sleep(0.1)
```

```
event = sense.stick.wait_for_event()
print("The joystick was {} {}".format(event.action, event.direction))
```

Das obige Beispiel sollte bei kurzer Joystickbetätigung zwei Ereignisse ausgegeben werden (1x gedrückt / 1x freigegeben). Man kann den Eingabepuffer löschen bevor auf ein neues Ereignis gewartet wird.

```
from time import sleep

sense = SenseHat()
event = sense.stick.wait_for_event()
print("The joystick was {} {}".format(event.action, event.direction))
sleep(0.1)
event = sense.stick.wait_for_event(emptybuffer=True)
print("The joystick was {} {}".format(event.action, event.direction))
```

get_events

Gibt eine Liste der InputEvent-Tupel zurück, die alle Ereignisse darstellen, die seit dem letzten Aufruf von `get_events` oder `wait_for_event` aufgetreten sind.

```
while True:
    for event in sense.stick.get_events():
        print("The joystick was {} {}".format(event.action, event.direction))
```

Das obige Beispiel zeigt den Status des Joysticks an, immer wenn ein Joystickereignis auftritt (Joystick in Richtung bewegen / Joystick in Ruhestellung) / Joystickposition gehalten / Joystick gedrückt.)

direction_up,
direction_left,
direction_right,
direction_down,
direction_middle,
direction_any

Diesen Attributen kann eine Funktion zugewiesen werden, die aufgerufen wird, wenn der Joystick in die zugehörige Richtung (oder in eine beliebige Richtung bei `direction_any`) gedrückt wird. Die zugewiesene Funktion darf entweder keinen Parameter oder einen einzelnen Parameter annehmen, der dem zugehörigen Ereignis übergeben wird.

```
from sense_hat import SenseHat, ACTION_PRESSED, ACTION_HELD, ACTION_RELEASED
from signal import pause

x = 3
```

```

y = 3

sense = SenseHat()

def clamp(value, min_value=0, max_value=7):
    return min(max_value, max(min_value, value))

def pushed_up(event):
    global y
    if event.action != ACTION_RELEASED:
        y = clamp(y - 1)

def pushed_down(event):
    global y
    if event.action != ACTION_RELEASED:
        y = clamp(y + 1)

def pushed_left(event):
    global x
    if event.action != ACTION_RELEASED:
        x = clamp(x - 1)

def pushed_right(event):
    global x
    if event.action != ACTION_RELEASED:
        x = clamp(x + 1)

def refresh():
    sense.clear()
    sense.set_pixel(x, y, 255, 255, 255)

sense.stick.direction_up = pushed_up
sense.stick.direction_down = pushed_down
sense.stick.direction_left = pushed_left
sense.stick.direction_right = pushed_right
sense.stick.direction_any = refresh
refresh()
pause()

```

Beachten Sie, dass das `direction_any` -Ereignis immer nach allen anderen Ereignissen aufgerufen wird, was es zu einem idealen Ereignis für Dinge wie das Aktualisieren der Anzeige macht (wie im obigen Beispiel).