"PROGRAMACIÓN EN EL EMULADOR DE SENSE HAT - RASPBERRY S.O."

Trabajo Preparatorio N°11
Laboratorio de Sistemas Embebidos

Melanny Cecibel Dávila Pazmiño
Ingeniería en Telecomunicaciones
Facultad de Eléctrica y Electrónica
Quito, Ecuador
melanny.davila@epn.edu.ec

***Abstract*—En el presente documento se presenta el sustento teórico para comprender el funcionamiento de Sense Hat al momento de usar Raspberry, con el fin de poder desarrollar aplicaciones amigables con el usuario.**

***Index Terms*—Raspberry, Python, Sense Hat, programación.**

I. Introducción

Para ampliar las capacidades de Raspberry se creó un shield especial llamado Sense Hat con varias capacidades para tomar y mostrar medidas e incrementar la interacción entre el usuario y la placa de desarrollo. Este shield, dadas sus capacidades y bajo coste ha sido usado ampliamente dentro de la comunidad [1].

El Sense HAT es una placa complementaria para Raspberry Pi, hecha especialmente para la misión Astro Pi (se lanzó a la Estación Espacial Internacional en diciembre de 2015) y actualmente está disponible para su libre adquisición.

El Sense HAT tiene una matriz de LED RGB de 8×8, un joystick de cinco botones e incluye los siguientes sensores:

- Giroscopio
- Acelerómetro
- Magnetómetro
- Temperatura
- Presión barométrica
- Humedad [1]

II. Objetivos

- Relacionar al estudiante con el emulador de la placa de desarrollo Sense HAT.
- Realizar un conjunto de scripts que permitan familiarizar al estudiante con la placa de desarrollo Sense HAT y la programación en Python [2].

III. Cuestionario

A. Consultar que es el lenguaje de programación Python y cuáles son las diferencias más relevantes entre las versiones de Python 2.7 y Python 3.0.

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con escritura dinámica y enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripts o pegamento para conectar componentes existentes. La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo de mantenimiento del programa [1]. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar están disponibles en formato fuente o binario sin cargo para todas las plataformas principales y pueden distribuirse libremente [3].

Diferencias entre Python 2.3 y Python 3.0

- **Función print:**La declaración de impresión de Python 2 ha sido reemplazada por la función print (), lo que significa que se debe envolver el objeto que quiere imprimir en paréntesis.Python 2 no tiene ningún problema con paréntesis adicionales, pero en contraste, Python 3 generaría un SyntaxError si se llama a la función de impresión sin los paréntesis.

```
print 'Python', python_version()
print 'Hello, World!'
print('Hello, World!')
print "text", ; print 'print more text on the same line'
```

```
Python 2.7.6
Hello, World!
Hello, World!
text print more text on the same line
```

Fig. 1. Print-Python versión 2

```
print('Python', python_version())
print('Hello, World!')

print("some text,", end="")
print(' print more text on the same line')
```

```
Python 3.4.1
Hello, World!
some text, print more text on the same line
```

Fig. 2. Print-Python versión 3

- **División de enteros:** En Python 2 la división entre números enteros es otro número entero, y para obtener un resultado con decimales el numerador o el denominador tiene que tener también al menos un decimal. Este cambio es particularmente peligroso si está transfiriendo código o si está ejecutando código Python 3 en Python 2, ya que el cambio en el comportamiento de división de enteros a menudo puede pasar desapercibido (no genera un error de sintaxis).

```
print 'Python', python_version()
print '3 / 2 =', 3 / 2
print '3 // 2 =', 3 // 2
print '3 / 2.0 =', 3 / 2.0
print '3 // 2.0 =', 3 // 2.0
```

```
Python 2.7.6
3 / 2 = 1
3 // 2 = 1
3 / 2.0 = 1.5
3 // 2.0 = 1.0
```

Fig. 3. División de enteros-Python versión 2

```
print 'Python', python_version()
print '3 / 2 =', 3 / 2
print '3 // 2 =', 3 // 2
print '3 / 2.0 =', 3 / 2.0
print '3 // 2.0 =', 3 // 2.0
```

```
Python 2.7.6
3 / 2 = 1
3 // 2 = 1
3 / 2.0 = 1.5
3 // 2.0 = 1.0
```

Fig. 4. División de enteros-Python versión 3

- **Unicode:** Python 2 tiene tipos ASCII str (), unicode () separado, pero ningún tipo de byte. Ahora, en Python 3, finalmente tiene cadenas Unicode (utf-8) y clases de 2 bytes: byte y arreglos de bytes.

```
print type(unicode('this is like a python3 str type'))
```

```
<type 'unicode'>
```

```
print type(b'byte type does not exist')
```

```
<type 'str'>
```

```
print 'they are really' + b' the same'
```

```
they are really the same
```

```
print type(bytearray(b'bytearray oddly does exist though'))
```

```
<type 'bytearray'>
```

Fig. 5. Unicode-Python versión 2

```
print('Python', python_version())
print('strings are now utf-8 \u03BCnico\u0394é!')
```

```
Python 3.4.1
strings are now utf-8 µnicoΔé!
```

```
print('Python', python_version(), end="")
print(' has', type(b' bytes for storing data'))
```

```
Python 3.4.1 has <class 'bytes'>
```

```
print('and Python', python_version(), end="")
print(' also has', type(bytearray(b'bytearrays')))
```

```
and Python 3.4.1 also has <class 'bytearray'>
```

Fig. 6. Unicode-Python versión 3

- **Función xrange:** El uso de xrange () es muy popular en Python 2.x para crear un objeto iterable, por ejemplo, en un bucle for o en una lista / conjunto-diccionario-comprensión.
  El comportamiento era bastante similar a un generador (es decir, ”evaluación perezosa”), pero aquí el xrange-iterable no es agotable, es decir, se puede iterar sobre él infinitamente.

Gracias a su “evaluación perezosa”, la ventaja del rango regular () es que xrange () es generalmente más rápido si tiene que iterar sobre él solo una vez (por ejemplo, en un bucle for). Sin embargo, a diferencia de las iteraciones de una sola vez, no se recomienda si repite la iteración varias veces, ya que la generación ocurre cada vez desde cero.
En Python 3, range () se implementó como la función xrange () para que una función xrange () dedicada ya no exista (xrange () genera un NameError en Python 3).

```
print 'Python', python_version()

print '\ntiming range()'
%timeit test_range(n)

print '\n\ntiming xrange()'
%timeit test_xrange(n)
```

```
Python 2.7.6

timing range()
1000 loops, best of 3: 433 µs per loop


timing xrange()
1000 loops, best of 3: 350 µs per loop
```

Fig. 7. range y xrange-Python versión 2

```
print('Python', python_version())

print('\ntiming range()')
%timeit test_range(n)
```

```
Python 3.4.1

timing range()
1000 loops, best of 3: 520 µs per loop
```

Fig. 8. range-Python versión 3

- **Iterar un diccionario:** Para iterar los elementos clave-valor de un diccionario se puede utilizar el método iteritems() o items() en Python 2. En Python 3 esta operación sólo puede realizarse con el método items(), ya que al usar iteritems() se tiene una excepción del tipo AttributeError.
  Del mismo modo, los métodos iterkeys() e itervalues() para iterar las claves y los valores de un diccionario respectivamente no existen en Python 3. En su lugar se debe usar los métodos keys() y values().

```
d = {'animal': 'perro', 'vehiculo': 'coche'}
for k,v in d.iteritems():
    print k,':',v
vehiculo : coche
animal : perro
```

Fig. 9. Iteración-Python versión 2

```
d = {'animal': 'perro', 'vehiculo': 'coche'}
for k,v in d.iteritems():
    print(k,':',v)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'dict' object has no attribute 'iteritems'
```

Fig. 10. Iteración-Python versión 3

- **Función next():** Recuperar el siguiente elemento de un iterador puede hacerse tanto con la función next() como con el método .next() en Python 2, mientras que en Python 3 next sólo puede utilizarse como función.

```
>>> iterador = (letra for letra in 'python')
>>> next(iterador)
'p'
>>> iterador.next()
'y'
```

Fig. 11. Función next-Python versión 2

```
>>> iterador = (letra for letra in 'python')
>>> next(iterador)
'p'
>>> iterador.next()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'generator' object has no attribute 'next'
```

Fig. 12. Función next-Python versión 3

- **Comparación de tipos:** Python 2 permite la comparación entre objetos de tipo distinto, sin embargo Python 3 es restrictivo en este aspecto dándonos una excepción del tipo TypeError.

```
>>> 1 < '1'
True
```

Fig. 13. Comparación de tipos-Python versión 2

```
>>> 1 < '1'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between instances of 'int' and 'str'
```

Fig. 14. Comparación de tipos-Python versión 3

- **Manejo de excepciones:** El manejo de excepciones ha cambiado ligeramente en Python 3. En Python 3 se debe usar la palabra clave ”as”.

```
print 'Python', python_version()
try:
    let_us_cause_a_NameError
except NameError, err:
    print err, '--> our error message'
```

```
Python 2.7.6
name 'let_us_cause_a_NameError' is not defined --> our error message
```

Fig. 15. Excepción-Python versión 2

```
print('Python', python_version())
try:
    let_us_cause_a_NameError
except NameError as err:
    print(err, '--> our error message')
```

```
Python 3.4.1
name 'let_us_cause_a_NameError' is not defined --> our error message
```

Fig. 16. Excepción-Python versión 3

B. Consultar los comandos y configuraciones necesarios para integrar el servicio de envío de correos SMTP en la tarjeta Raspberry Pi. Considerar la utilización de la versión de sistema operativo Raspbian 9.13 o superior. Realizar un script en Python que permita el envío de un correo electrónico desde su cuenta personal de Gmail a su cuenta Institucional. Agregar las líneas de código empleadas en el contenido del Trabajo Preparatorio.

SMTP son las siglas en inglés para “protocolo simple de transferencia de correo”. SMTP es el protocolo estándar para proporcionar servicios de correo electrónico en una red TCP/IP. Este servidor ofrece la posibilidad de recibir y enviar mensajes de correo electrónico.

SMTP es un protocolo de capa de aplicación que proporciona la entrega y transmisión de correo electrónico a través de Internet [4]. A continuación se presenta el script que permite desarrollar lo solicitado:

```
# Escuela Politecnica Nacional
# CP-Sistemas Embebidos
# Practica 11-Literal 2
#Importacion de las librerias email.mime.multipart y email.mime.text
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import smtplib

# Creacion del objeto de mensaje
msg = MIMEMultipart()

# Parametros del mensaje
msg['From'] = input('Ingrese el correo de donde desea enviar: ')
clave= input('Ingrese la contrasena del correo: ')
msg['To'] = input('Ingrese el correo del destinatario: ')
msg['Subject'] = input('Coloque el asunto del correo: ')
mensaje= input('Inserte el mensaje: ')

#Agregacion en el cuerpo del correo
msg.attach(MIMEText(mensaje, 'plain'))

# Creacion del servidor
server = smtplib.SMTP('smtp.gmail.com: 587')
server.starttls()

# Ingreso de las credenciales para enviar el e-mail
server.login(msg['From'], clave)

# Envio del mensaje mediante el servidor
server.sendmail(msg['From'], msg['To'], msg.as_string())
#Desconexion del servidor
server.quit()
#Mensaje de confirmacion
print("Correo enviado con exito")
```

```
Ingrese el correo de donde desea enviar: mcdp1101@gmail.com
Ingrese la contrasena del correo: 1234mcdp
Ingrese el correo del destinatario: melannycdp@hotmail.com
Coloque el asunto del correo: Comprobacion
Inserte el mensaje: Correo enviado desde raspberry mediante el uso de SMTP
Correo enviado con exito
```

Fig. 17. Uso del código desarrollado

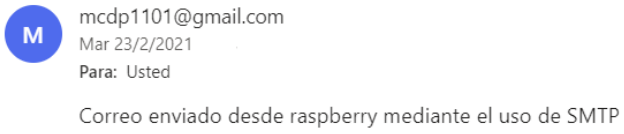

Fig. 18. Comprobación del correo recibido

C. Consultar que es la placa de desarrollo Sense HAT, describir detalladamente sus componentes y describir los comandos necesarios para incorporar de manera física con la tarjeta Raspberry Pi.

Sense HAT es una placa complementaria para Raspberry Pi, permite realizar mediciones de temperatura, humedad, presión y orientación, y generar información utilizando su matriz LED incorporada [5].

- **Sensores presentes en la placa:**
  - Giroscopio
  - Acelerómetro
  - Magnetómetro
  - Temperatura
  - Presión barométrica
  - Humedad [6]

- **Comandos previos a su uso:**

```
sudo apt-get update
sudo apt-get install sense-hat
sudo pip-3.2 install pillow
```

Script 1. Comandos para la instalación de Sense Hat mediante el terminal

- **Libería senseHat:** Una biblioteca o librería es el código fuente que en este caso proporciona todo lo que se necesita para hacer uso de las funciones de la placa, primero se debe incluir la biblioteca de Python para Sense HAT al comienzo de la secuencia de comandos. A continuación se presentan las dos líneas comandos para agregar dicha librería:

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

Script 2. Comandos para importar la libería SenseHat

- **Pantalla LED:** A través de ella se puede mostrar texto, imágenes, símbolos, etc. Con los siguientes comandos se podrá mostrar el mensaje "Hola" en dicha pantalla [6].

```
from sense_hat import SenseHat
sense = SenseHat()
sense.show_message("Hola")
```

Script 3. Ejemplo de envío de mensaje

En el programa anterior se puede cambiar la velocidad con la que se desplaza el texto horizontalmente, su color, y el color de fondo con los siguientes parámetros:

  - scroll_speed: Por defecto 0.1 , como mayor sea el número menor será la velocidad.
  - text_colour: Se compone de una combinación de rojo, verde y azul que pueden ser combinado, con valores de 0 a 255 cada uno de ellos.
  - back_colour: Para formar el color de fondo de la pantalla, por defecto negro, de igual que ocurre que con text_colour se compone de una combinación de los 3 colores básicos (rojo, verde y azul) [5].

- **Sensor IMU:** El sensor IMU (unidad de medida inercial) es una combinación de tres sensores, cada uno con un eje x, y y z. Por esta razón, se considera un sensor de 9 dof (grados de libertad).
  1) Giroscopio
  2) Acelerómetro
  3) Magnetómetro (brújula)

  Esta API le permite utilizar estos sensores en cualquier combinación para medir la orientación o como sensores individuales por derecho propio [6].
- **Comando set_imu_config:** Habilita y deshabilita la contribución del giroscopio, acelerómetro y/o magnetómetro a las siguientes funciones de orientación.

TABLA I
PARÁMETROS CONFIGURABLES EN EL COMANDO SET_IMU_CONFIG

Parámetro	Tipo	Valores Válidos	Explicación
compass_enabled	Booleano	Verdadero, falso	Si la brújula debe estar habilitada o no.
gyro_enabled	Booleano	Verdadero, falso	Si el giroscopio debe estar habilitado o no.
accel_enabled	Booleano	Verdadero, falso	Si el acelerómetro debe estar habilitado o no.

```
from sense_hat import SenseHat

sense = SenseHat()
sense.set_imu_config(False, True, False)  # gyroscope only
```

Fig. 19. Ejemplo del comando set_imu_config

- **Comando get_orientation_radians:** Obtiene la orientación actual en radianes utilizando los ejes principales de cabeceo (pitch), balanceo (roll) y guiñada (yaw) de la aeronave [7].

```
from sense_hat import SenseHat

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

# alternatives
print(sense.orientation_radians)
```

Fig. 20. Ejemplo del comando get_orientation_radians

- **Comando get_orientation_degrees:** Obtiene la orientación actual en grados utilizando los ejes principales de cabeceo (pitch), balanceo (roll) y guiñada (yaw) de la aeronave [7].

```
from sense_hat import SenseHat

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

# alternatives
print(sense.orientation_radians)
```

Fig. 21. Ejemplo del comando get_orientation_degrees

D. Consultar y describir la incorporación de la librería SenseHat los comandos: sense.show_message(), sense.show_letter(), sense.set_pixel() y sense.set_rotation(). Añadir los objetos de configuración de cada uno de los comandos.

- **Comando sense.show_message():** Desplaza un mensaje de texto de derecha a izquierda a través de la matriz de LED y a la velocidad especificada, en el color especificado y el color de fondo [6].

TABLA II
PARÁMETROS DE CONFIGURACIÓN

Parámetro	Tipo	Valores Válidos	Explicación
text_string	Cadena de caractéres	Cualquier texto	Muestra el mensaje a desplazar.
scroll_speed	Flotante	Cualquier número flotante.	Es la velocidad de desplazamiento del texto, tiempo de pausa entre el desplazamiento a la derecha, su valor por defecto es de 0,1.
text_colour	Lista	[R, G, B]	Lista que contiene el color del texto, por defecto el color es blanco [255,255,255].
back_colour	Lista	[R, G, B]	Lista que describe el color de fondo, por defecto es negro es decir las tres variables son 0.

```
from sense_hat import SenseHat

sense = SenseHat()
sense.show_message("One small step for Pi!", text_colour=[255, 0, 0])
```

Fig. 22. Comando sense.show_message()

- **Comando sense.show_letter():** Muestra un solo carácter de texto en la matriz de LED [8].

TABLA III
PARÁMETROS PARA CONFIGURAR

Parámetro	Tipo	Valores Válidos	Explicación
s	Cadena de caractéres	Una cadena de texto de longitud 1.	Letra para mostrar.
text_colour	Lista	[R, G ,B]	Lista que contiene el color de la letra, cada elemento debe tener un número entre 0 y 255 por defecto el color es blanco.
bakc_colour	Lista	[R, G, B]	Lista que contiene el color de fondo, por defecto es negro [0,0,0].

```
import time
from sense_hat import SenseHat

sense = SenseHat()

for i in reversed(range(0,10)):
    sense.show_letter(str(i))
    time.sleep(1)
```

Fig. 23. Comando sense.show_letter()

- **Comando sense.set_pixel():** Actualiza toda la matriz de LED basándose en una lista de valores de píxeles de 64 longitudes, permitiendo crear figuras para mostrar en la pantalla LED [9].

TABLA IV
PARÁMETRO DE CONFIGURACIÓN

Parámetro	Tipo	Valores Válidos	Explicación
pixel_list	List	[[R, G, B]*64]	Contiene una lista conformada por 64 sublistas de [R, G, B] píxeles(rojo, verde, azul). Cada elemento debe ser un número debe ser un número entero entre 0 y 255.

```
from sense_hat import SenseHat

sense = SenseHat()

X = [255, 0, 0]  # Red
O = [255, 255, 255]  # White

question_mark = [
O, O, O, X, X, O, O, O,
O, O, X, O, O, X, O, O,
O, O, O, O, O, X, O, O,
O, O, O, O, X, O, O, O,
O, O, O, X, O, O, O, O,
O, O, O, X, O, O, O, O,
O, O, O, O, O, O, O, O,
O, O, O, X, O, O, O, O
]

sense.set_pixels(question_mark)
```

Fig. 24. Comando sense.set_pixel()

- **Comando sense.set_rotation():** Se puede usar esta función para corregir la orientación de la imagen que se muestra en la pantalla LED [9].

TABLA V
PARÁMETROS DE CONFIGURACIÓN

Parámetro	Tipo	Valores Válidos	Explicación
r	Entero	0, 90, 180 y 270	Describe el ángulo para rotar a la matriz LED.
redraw	Booleano	Verdadero, Falso	Su uso consiste en si se debe volver a dibujar o no lo que ya se muestra en la matriz del LED, por defecto su valor es verdadero.

```
from sense_hat import SenseHat

sense = SenseHat()
sense.set_rotation(180)
# alternatives
sense.rotation = 180
```

Fig. 25. Comando sense.set_rotation

E. Consultar la funcionalidad y objetos de configuración de los comandos: get_temperature(), get_pressure() y get_humidity().

- **Función get_temperature:** Obtiene la temperatura actual en milibares del sensor de presión [9].

```
from sense_hat import SenseHat

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

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

Fig. 26. Código fuente de la función get_temperature

  - Función get_temperature_from_humidity: Obtiene la temperatura actual en grados Celsius del sensor de humedad.

```
from sense_hat import SenseHat

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

Fig. 27. Código fuente de la función get_temperature_from_humidity

  - Función get_temperature_from_pressure: Obtiene la temperatura actual en grados Celsius del sensor de presión [9].

```
from sense_hat import SenseHat

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

Fig. 28. Código fuente de la función get_temperature_from_pressure

- **Función get_humidity:** Obtiene el porcentaje de humedad relativa del sensor de humedad.

```
from sense_hat import SenseHat

sense = SenseHat()
humidity = sense.get_humidity()
print("Humidity: %s %%rH" % humidity)

# alternatives
print(sense.humidity)
```

Fig. 29. Código fuente de la función get_humidity

- **Función get_pressure:** Obtiene la presión actual en milibares del sensor de presión [9].

```
from sense_hat import SenseHat

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

# alternatives
print(sense.pressure)
```

Fig. 30. Código fuente de la función get_pressure

F. Realizar un script en Python usando la placa Sense Hat que permita a través del arreglo de leds de 8x8 presentar el mensaje. "Sistemas Embebidos_Grupox_Apellido_Nombre". Donde x representa el grupo al que pertenecen. Agregar las líneas de código empleadas en el contenido del Trabajo Preparatorio.

```
#Escuela Politecnica Nacional
#CP-Sistemas Embebidos
#Practica Nro. 11- literal 6

#Importacion de la libreria SenseHat
from sense_emu import SenseHat
#Creacion del objeto sense
sense=SenseHat()
#Lazo while para mostrar el mensaje constantemente
while 1:
    #Presentacion del mensaje solicitado
    sense.show_message("Sistemas
    Embebidos_Grupo4_Davila_Melanny",scroll_speed
    =0.17,text_colour=[255,26,140],back_colour
    =[255,213,234])
    #Modificacion del color del texto y del fondo
```

Script 4. Implementación solicitada

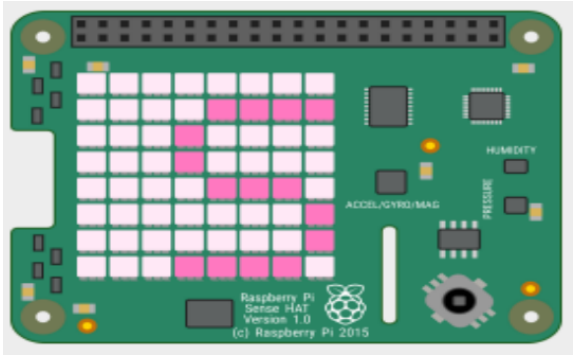

Fig. 31. Impresión del mensaje solicitado

REFERENCES

[1] T. R. P. Foundation, "Buy a Sense HAT", Raspberry Pi. https://www.raspberrypi.org/products/sense-hat/ (accedido feb. 20, 2021).

[2] E. Espinosa, E. Tatayo, "PROGRAMACIÓN EN EL EMULADOR DE SENSE HAT - RASPBERRY S.O.". C.P. SISTEMAS EMBEBIDOS, Accedido: feb. 16, 2021. [En línea].

[3] “What is Python? Executive Summary”, Python.org. https://www.python.org/doc/essays/blurb/ (accedido feb. 20, 2021).
[4] “Diferencias entre Python 2 y 3”, Programa en Python, ene. 26, 2019. https://www.programaenpython.com/miscelanea/diferencias-entre-python-2-y-3/ (accedido feb. 21, 2021).
[5] “How to Send SMTP Email using Raspberry Pi”. https://www.iotdesignpro.com/projects/sending-smtp-email-using-raspberry-pi (accedido feb. 22, 2021).
[6] “SENSE HAT Raspberry Pi — Placas de desarrollo, kits, programadores — DigiKey”. https://www.digikey.es/product-detail/es/raspberry-pi/SENSE-HAT/1690-1013-ND/6196429 (accedido feb. 22, 2021).
[7] D. P. E. Team, “Sense HAT For Raspberry Pi, 6 Types of Sensors”, Device Plus, jun. 20, 2017. https://www.deviceplus.com/raspberry-pi/the-sense-hat-add-on-board-for-raspberry-pi-6-types-of-sensors/ (accedido feb. 22, 2021).
[8] “Raspberry Sense Hat – Astro Pi – 1 – Pantalla y Sensores de Ambiente”, Alteageek, tutoriales, raspberry pi y cisco, en español, jul. 09, 2016. https://alteageek.com/2016/07/09/raspberry-sense-hat-astro-pi/ (accedido feb. 22, 2021).
[9] “API Reference - Sense HAT”. https://pythonhosted.org/sense-hat/api/#get_temperature (accedido feb. 22, 2021).