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

Informe N°11
Laboratorio de Sistemas Embebidos

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

2nd Jonathan Álvarez
Ingeniería en Telecomunicaciones
Facultad de Elétrica y Eléctronica
Quito, Ecuador
jonathan.alvarez@epn.edu.ec

***Abstract*—En el siguiente informe se presentarán los beneficios que tiene la placa de desarrollo Sense Hat y sus aplicaciones en la industria 4.0. Seguido se presentara un script en python que simula el juego Snake. Se hablará de los proyectos de software de codigo abierto YateBTS y OpenAirInterface además de los comandos necesarios para que raspberry Pi establezca comunicacion con un dispositivo SDR para finalmente describir los comandos para que establezca comunicacion con una estación movil.**

***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. Analizar los beneficios de la placa de desarrollo Sense HAT y describir de manera detallada tres aplicaciones en la Industria 4.0.

Sense Hat es un dispositivo que sirve para aprender y hacer proyectos con ella. El tamaño de esta placa es bastante reducido, sin embargo, su diseño es excelente dado a la cantidad de implementaciones que se puede realizar con esta placa. Se puede mostrar la temperatura y humedad. Ver dónde está el norte, aunque esto requiere una calibración especial del sensor de campo magnético y un sin fin de posibilidades más [3].

- **Estación climática:** Por medio del uso de los sensores tanto de humedad como de temperatura se plantea crear una estación climática que por medio del arreglo de LEDs 8x8 muestre si la temperatura aumenta o disminuye por medio de caracteres representativos. La forma de presentar los datos es por medio de una flecha roja apuntando hacia arriba cuando la temperatura aumentó con respecto a la última lectura, una flecha azul apuntando hacia abajo cuando la temperatura disminuyó, y azul y roja barras(como un extraño signo igual) cuando la temperatura permanece igual entre mediciones. El proyecto es realmente fáacil de completar. Simplemente ensamblar el hardware (que demora aproximadamente 5 minutos), instalar algunas bibliotecas de Python, descargar y configurar el código del proyecto y listo [4].

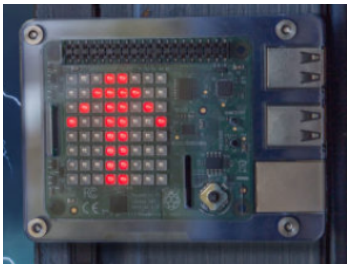

Fig. 1. Implementación

- **Brújula:** Haciendo uso de una batería interna y de la placa de desarrollo Sense Hat se puede presentar una brújula por medio del arreglo de LEDs 8x8 y ajustar sus valores por medio del acelerómetro y el sensor magnético. La dificultad de este proyecto yace en la programación de la toma de datos por medio del sensor magnético y en el diseño del contenedor de todos los elementos del circuito.

Fig. 2. Inmplementación

- **Control término de una habitación:** Para poder implementar esta aplicación se requiere de una placa Raspberry Pi y una placa de desarrollo Sense Hat,lo que se pretende es, por medio del sensor de temperatura de la placa Sense Hat, controlar los valores máximos y mínimos de temperatura que existen en una habitación, la programación es muy sencilla ya que solo se requiere medir la temperatura con un script de Python donde solo se haga uso de la librería de Sense Hat y de la librería de tiempo para agregar retardos al reconocimiento de temperatura según se requiera [4]. Una vez se hayan detectado los límites de temperatura se activará por medio de los pines GPIO de la placa un dispositivo calefactor o de aire acondicionado según se requiera [3].

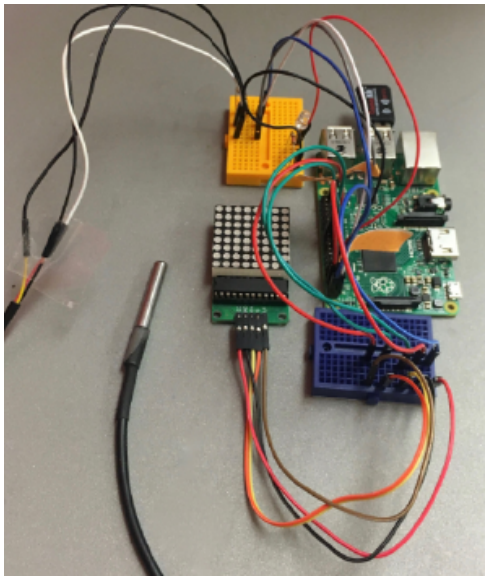

Fig. 3. Circuito implementado

B. Realizar un script en python que permita simular el juego "Snake", el cual puede obtener una puntuación máxima de 50 puntos y que se despliegue en una pantalla de 480 x 240 pixeles. Se recomienda trabajar con las librerías tkinter o turtle. Añadir el código empleado en el contenido del Informe.

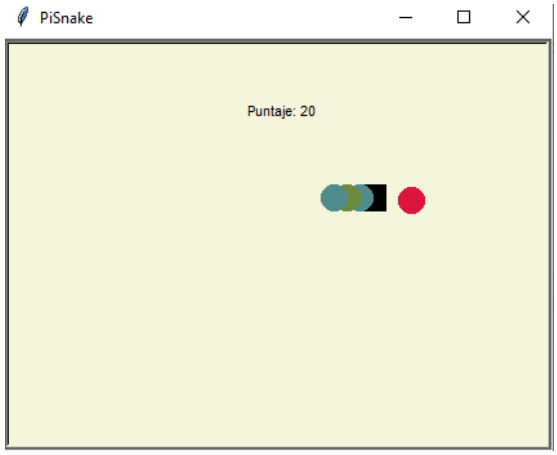

Fig. 4. Pantalla del juego snake

```
# CP Sistemas Embebidos
# Se importa las librerias correspondientes
import turtle
import time
import random
# Se definen parametros de retraso y velocidad
delay = 0.1
velocidad = 10
cuerpo = []
# Se da tamanio y color a la ventana
vent = turtle.Screen()
vent.title("PiSnake")
vent.bgcolor("Beige")
vent.setup(width=480, height=240) # Se define el
    tamanio de fondo
vent.tracer(0)
# Se define a la serpiente y se dejan los valores
    por defecto
cabeza = turtle.Turtle()
cabeza.speed(0)
cabeza.shape("square")
cabeza.color("Black")
cabeza.penup()
cabeza.goto(0, 0)
cabeza.direction = "right"
# Se define los parametros de la comida, su
    velocidad
comida = turtle.Turtle()
comida.speed(0)
comida.shape("circle")
comida.color("Crimson")
comida.penup()
comida.goto(30, 40)
# Se trabaja sobre el puntaje del juego
puntaje = 0
pun = turtle.Turtle()
pun.speed(0)
pun.shape("square")
pun.color("black")
pun.penup()
pun.hideturtle()
pun.goto(0, 100)
```

```
pun.write("Puntaje: " + str(puntaje), align="center")
# Funciones que responden a los eventos del teclado
def subir():
    cabeza.direction = "up"

def bajar():
    cabeza.direction = "down"

def dere():
    cabeza.direction = "right"

def izq():
    cabeza.direction = "left"

def mov():
    if cabeza.direction == "up":
        y = cabeza.ycor()
        cabeza.sety(y + velocidad)

    if cabeza.direction == "down":
        y = cabeza.ycor()
        cabeza.sety(y - velocidad)

    if cabeza.direction == "left":
        x = cabeza.xcor()
        cabeza.setx(x - velocidad)

    if cabeza.direction == "right":
        x = cabeza.xcor()
        cabeza.setx(x + velocidad)

# Se configura los eventos que causaran las acciones en
vent.listen()
vent.onkeypress(subir, "Up")
vent.onkeypress(bajar, "Down")
vent.onkeypress(dere, "Right")
vent.onkeypress(izq, "Left")

# Variable auxiliar de la comida
cComida = 0
# Desarrollo del juego
while True:
    vent.update()
    # Definicion del borde x
    if cabeza.xcor() > 240 or cabeza.xcor() < -240:
        cabeza.goto((-1 * cabeza.xcor()), cabeza.ycor())
    # Definicion del borde y
    if cabeza.ycor() > 110 or cabeza.ycor() < -110:
        cabeza.goto(cabeza.xcor(), (-1 * cabeza.ycor()))

    if cabeza.distance(comida) < 20:
        # COndiciones de crecimiento
        x = random.randint(-210, 210)
        y = random.randint(-100, 100)
        comida.goto(x, y)
        cComida = cComida + 1
        puntaje = puntaje + 5
        # Se agrega a la cola
        nSeg = turtle.Turtle()
        nSeg.speed(0)
        nSeg.shape("circle")
        if cComida % 2:
            nSeg.color("DarkOliveGreen4")
        else:
            nSeg.color("DarkSlateGray4")
        nSeg.penup()
        cuerpo.append(nSeg)

    # Condiciones cuando se pierde
    for seg in cuerpo:
        if seg.distance(cabeza) < 10:
            time.sleep(5)
            cabeza.goto(0, 0)
            cabeza.direction = "stop"
            for cseg in cuerpo:
                cseg.goto(10000, 10000)
            cuerpo.clear()
            puntaje = 0
    if len(cuerpo) > 0:
        x = cabeza.xcor()
        y = cabeza.ycor()
        cuerpo[0].goto(x, y)

    for i in range(len(cuerpo) - 1, 0, -1):
        x = cuerpo[i - 1].xcor()
        y = cuerpo[i - 1].ycor()
        cuerpo[i].goto(x, y)
    pun.write("Puntaje: " + str(puntaje), align="center")
    mov()
    time.sleep(delay)
    pun.clear()

vent.mainloop()
```

Script 1. Codigo implementado para el juego snake

C. Consultar y describir que son los proyectos open source de software YateBTS y OpenAirInterface.

- **YateBTS:** Es una implementación de software de una red de acceso por radio GSM/GPRS basada en Yate y es compatible con las redes centrales GSM/GPRS SS7 MAP y LTE IMS integradas en un servidor de red central unificado YateUCN.
  YateBTS ofrece un enfoque único, diferente al de las redes tradicionales de acceso por radio, con mayor flexibilidad, escalabilidad y se puede actualizar fácilmente con nuevas funciones [5].
  Funcionamiento: Cuando su usa un celular en una red YateBTS, la señal GSM llega a la antena de YateBTS. Posteriormente, la señal pasa a las Capas 1 y 2 donde se procesa la señal GSM y se alimenta a través del enchufe a Yate. Yate procesa la conexión recibida con el protocolo necesario (SIP u otro) para comunicarse con el servidor externo del proveedor de VoIP, por ejemplo, que lo vincula a la persona o máquina con la que desea comunicarse.
  YateBTS ha sido creado con el propósito de ofrecer una solución mejorada y combinada entre L1 PHY, L2 Link Layer y L3 Radio Resource Manager, también conocido como MBTS, y las múltiples funciones de Yate, como IAX por satélite, SS7 y Diameter, USSD, RManager, roaming o conmutación de teléfono local [5].
  YateBTS es un elemento fundamental en YateUCN, una solución de un solo núcleo para redes LTE y 2/2.5G. Por lo general, está configurado para funcionar en uno de dos modos:
  - Modo de red en una PC (NiPC): en este modo, YateBTS actúa como una red GMS/GPRS autónoma, que se conecta al mundo exterior a través de protocolos VoIP y/o ISDN.

  - Modo Radio Access Network (RAN): en este modo, YateBTS actúa como un elemento de una red GSM/GPRS de múltiples sitios más grande. Las funciones básicas de la red las proporciona un servidor externo. Este servidor externo puede ser cualquiera de los siguientes:
    * YateUCN para redes de operación móvil 2G SS7/MAP o 4G IMS/VoLTE o
    * otra instalación de YateBTS en modo NiPC para redes privadas de múltiples sitios.

  Antes de pasar a otros conceptos, es importante analizar las razones principales por las que YateBTS es diferente a cualquier estación base 2G típica del mercado.
- **OpenAirInterface:** Es una plataforma abierta de experimentación y prototipado creada por el Departamento de Comunicaciones Móviles de EURECOM para permitir la innovación en el área de redes y comunicaciones móviles/inalámbricas. Con OAI, la funcionalidad del transceptor (de una estación base, punto de acceso, terminal móvil, red central, etc.) se realiza a través de una interfaz de radio de software conectada a una computadora host para procesar; este enfoque es similar a otros plataformas de creación de prototipos de radio definida por software (SDR) ampliamente utilizadas en la comunidad de investigación de redes inalámbricas como SORA [6].
  La plataforma OpenAirInterface ofrece un sistema 4G basado en un conjunto de programas de software. Estos programas pueden ser probados y modificados individualmente por las empresas usuarias, independientemente de los demás programas. El objetivo es establecer las nuevas características de lo que se convertirá en la red 5G.
  Funcionamiento:
  El objetivo es implementar los componentes de software necesarios para un sistema 4G completo. Esto implica el módem de un terminal móvil, el software para estaciones de retransmisión de radio, así como el software para los enrutadores específicos utilizados para un núcleo de red. Por tanto, el software se ocupa de todos los procesos que intervienen en la capa de radio (modulación, codificación, etc.) de los protocolos de comunicación. Se ejecuta en los procesadores Intel x86 que se encuentran en PC y grupos de computadoras. Esto significa que es compatible con los desarrollos en la nube. Para instalarlo, debe tener una tarjeta de radio conectada a la PC, que sirve como terminal, y una segunda PC, que sirve como estación de relevo [7].
  A continuación, dependiendo de lo que se necesite hacer, se puede tomar solo una parte de la implementación del software. Por ejemplo, utilizar terminales móviles comerciales y conectarnos a una red compuesta por un relé OpenAirInterface y un núcleo de red comercial. Cualquier combinación es posible. Por lo tanto, se ha establecido una cadena de red completa para 4G, que puede avanzar hacia la red 5G utilizando todos estos programas de software [8].

  Características:
  - Una implementación de software de código abierto del sistema celular móvil de cuarta generación (4G) que es totalmente compatible con los estándares 3GPP LTE y se puede utilizar para la experimentación y demostración en tiempo real en puertas/exteriores.
  - Capacidad de emulación incorporada que se puede utilizar dentro del mismo entorno de ejecución real para una transición sin problemas entre experimentación real y emulación repetible y escalable. Específicamente, se admiten dos modos de emulación de capa física (PHY) que difieren en el nivel de detalle en el que se encuentra capa PHY (física).

D. Consultar y describir los comandos necesarios para que la tarjeta Raspberry Pi establezca comunicación con un dispositivo SDR y con el software YateBTS para realizar una estación móvil GSM/GPRS.

Para la realización de lo solicitado se ha empleado una Raspberry Pi 3 con la distribución Raspbian Buster Lite. Instalar los drivers y librerias necesarias para usar el dispositivo SDR. Para ello ejecutamos el siguiente comando:

```
sudo apt install rtl-sdr librtlsdr-dev
```

Una vez instalado el dispositivo se dará permisos a los usuarios no root para poder usar el dispositivo. Para ello se debe seguir los siguientes pasos para crear una regla udev [9].

1) Con el dispositivo SDR conectado, ejecutamos el siguiente comando para obtener la lista de dispositivos USB conectados:

```
lsusb
```

2) Buscar en la lista el dispositivo SDR y observar los identificadores de fabricante y de dispositivo. En este caso son 0bda y 2838 respectivamente.

```
Bus 001 Device 004: ID 0bda:2838 Realtek
    Semiconductor Corp. RTL2838 DVB-T
```

3) Creación de un archivo de reglas nuevo:

```
sudo nano /etc/udev/rules.d/rtl-sdr.rules
```

4) Añadir la siguiente regla modificando los identificadores si fuese necesario para que correspondan con los obtenidos en el paso anterior:

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0bda",
    ATTRS{idProduct}=="2838", MODE:="0666"
```

5) Por último se desconecta el dispostivo SDR y se lo vuelve a conectar para que se aplique la regla y se ejecuta el siguiente comando para comprobar que el dispositivo funciona correctamente:

```
rtl_test
```

Si todo fue correctamente realizado se presenta la siguiente salida:

```
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 82020715

Using device 0: Generic RTL2832U OEM
Detached kernel driver
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7
     7.7 8.7 12.5 14.4 15.7 16.6 19.7 20.7 22.9
     25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6
    40.2 42.1 43.4 43.9 44.5 48.0 49.6
[R82XX] PLL not locked!
Sampling at 2048000 S/s.

Info: This tool will continuously read from the
     device, and report if
samples get lost. If you observe no further
    output, everything is fine.

Reading samples in async mode...
Allocating 15 zero-copy buffers
lost at least 140 bytes
```

Para utilizar YateBTS para realizar una estación móvil GSM/GPRS se realizan los siguientes pasos:

1) Instalar las bibliotecas BladeRF

```
sudo apt install bladerf
sudo apt install libbladerf-dev
sudo apt install bladerf-fpga-hostedx40
sudo apt  autoremove
sudo addgroup yate
sudo usermod -a -G yate pi
sudo apt -y install git telnet apache2
sudo apt install php libapache2-mod-php -y
sudo apt install cmake automake
sudo apt install libusb-1.0-0-dev
sudo apt install subversion
```

2) Con todas las bibliotecas BladeRF instaladas, conecte BladeRF a un puerto USB en Raspberry PI y emita el siguiente comando:

```
dmesg
```

El comando dmesg muestra que BladeRF está conectado como un dispositivo USB en la Raspberry PI [11].

3) Se procede a descargar fuentes de Yate de SVN. Una vez que tenga el cliente SVN instalado, obtener las fuentes es simple, se debe escribir el siguiente comando en su terminal:

```
svn co http://voip.null.ro/svn/yate/trunk yate
```

4) Instalar Yate desde SV: ir al directorio donde se descargó Yate y ejecute autogen.sh. Esto generará el script de configuración que verifica las dependencias. Luego ejecute make install-noapi que compilará e instalará Yate. En lugar de make install-noapi, puede usar make install, pero asegúrese de instalar el paquete doxygen o kdoc [10].

```
cd yate/
./autogen.sh
./configure
sudo make install-noapi
```

5) Compilar Yate:

```
./autogen.sh
./configure
sudo make
sudo make install
```

- La primera línea generará el archivo de configuración
- La segunda línea configurará el código fuente
- La tercera línea compilará el código fuente.

6) Descargar las fuentes de YateBTS en cualquier directorio, en este escenario es el directorio de inicio igual que Yate.

```
svn co http://voip.null.ro/svn/yatebts/trunk
    yatebts
```

7) Asegúrese de tener solo una versión de los componentes de desarrollo de Yate.

```
which -a yate-config
/usr/local/bin/yate-config
```

8) Si ve más de una instancia de yate-config, debe desinstalar todas menos una. Puede utilizar la opción –version para comprobar qué versión es cada una:

```
which -a yate-config
/usr/local/bin/yate-config
/usr/bin/yate-config
/usr/local/bin/yate-config --version
6.2.1
/usr/bin/yate-config --version
5.2.0
```

9) Compile e instale YateBTS de la siguiente manera:

```
sudo hacer instalar
sudo ldconfig
```

Después de esto, los módulos, scripts y configuraciones de YateBTS se moverán a los directorios apropiados donde se encuentran otros componentes similares de Yate.

10) Cree un enlace simbólico a la GUI web de NIPC en la carpeta Apache WWW y otorgue permiso de escritura a los archivos de configuración.

```
cd /var/www/html
/var/www/html $ sudo ln -s
/usr/local/share/yate/nipc_web nipc
/var/www/html $ sudo chmod -R a+w
/usr/local/etc/yate
```

11) Para la operatividad GSM BTS, debe configurar la banda de radio, la frecuencia de operación, el código de país, el código de operador y la potencia de la radio. Para no perturbar las frecuencias de ningún operador, seleccione la banda de radio GSM850, con la disponibilidad de dos SIM gratuitas de otro país con código de país (629) y código de operador (02) para conexión automática a la red, el código de país estándar y el código de operador para que se utilizarán para las pruebas son MCC = 001, MNC = 01 pero en este caso el móvil se registrará manualmente [11]. Entonces los valores serán los siguientes:

```
Radio.Band=GSM850
Radio.C0=132
Identity.MCC=629
Identity.MNC=02
Radio.PowerManager.MaxAttenDB=40
Radio.PowerManager.MinAttenDB=40
```

E. Consultar y describir los comandos necesarios para que la tarjeta Raspberry Pi establezca comunicación con un dispositivo SDR y con el software OpenAirInterface para realizar una estación móvil LTE.

Se debe actualizar la Raspberry Pi usando los siguientes comandos:

```
sudo apt update -y
sudo apt dist-upgrade -y
sudo rpi-update
```

Script 2. Comandos de actualización

Se instalan las librerias necesarias

```
sudo apt install libqmi-utils
sudo apt install udhcpc
```

Script 3. Intalacion librerias

Permiten interactuar con modems basados en Qualcom Actuvamos la comunicación UART con el dispositivo

```
sudo raspi-config
```

Script 4. Ingresar a config

Nos dirigirá a las opciones debemos elegir la opcion 3. Interface Options luego a la P6 Serial Port y seleccionamos No para la primera pregunta y si para la siguiente Para configurar el modulo SIM7600 ejecutamos los siguientes comandos

```
sudo qmicli -d /dev/cdc-wdm0 --dms-set-operating-
    mode='online'
```

Script 5. Verificar si esta online

para verificar si esta en linea

```
sudo qmicli -d /dev/cdc-wdm0 --dms-get-operating-
    mode
```

Script 6. modo de operacion

para verificar la fuerza de la señal y su calidad

```
sudo qmicli -d /dev/cdc-wdm0 --nas-get-signal-
    strength
```

Script 7. Calidad de la señal

Para ver el nombre de la operadora

```
sudo qmicli -d /dev/cdc-wdm0 --nas-get-home-network
```

Script 8. Nombre de la operadora

Para conectarse a la red movil usamos los siguientes comandos:

```
sudo qmicli -p -d /dev/cdc-wdm0 --device-open-net='
    net-raw-ip|net-no-qos-header' --wds-start-
    network="apn='YOUR_APN',auth=BOTH,username='
    YOUR_USERNAME',password='YOUR_PASSWORD',ip-type
    =4" --client-no-release-cid
```

Script 9. Acceso a la red movil

F. Conclusiones:

Jonathan Álvarez

- La placa Sense Hat agrupa una gran cantidad de sensores y dispositivos que permiten ampliar las capacidades de Raspberry Pi sin la necesidad de tener varios componente. También esta placa permite desarrollar aplicaciones rápidas y que son dinámicas además de mas serias que permitió que esta placa se prueba en la ISS
- Existe gran cantidad de documentación y acceso a proyectos ya creados, además de foros de preguntas y respuestas de la comunidad de desarrollo en torno a Raspberry Pi lo que permite el desarrollo mas fácil de nuestros proyectos además del uso del lenguaje de programación Python que tiene mucha popularidad

Melanny Dávila

- Mediante el uso de la plaza Sense Hat, se puede implementar una gran cantidad de aplicaciones amigables con el usuario, sobretodo aquellas que buscan realizar el análisis de fenómenos meteorológicos. Además dado que existe gran cantidad de repositorios, la implementación de aplicaciones es cada vez más fácil.
- Mediante el correcto uso de esta placa se puede optimizar muchos procesos y a su vez se puede crear una pequeña base de datos casera donde se puede almacenar los valores recogidos durante un periodo de tiempo.
- Gracias al uso de protocolos de transporte como SMTP conjuntamente con la placa Raspebrry Pi se puede realizar aplicaciones mucha más realistas como lo es el envío de correos electrónicos.

G. Recomendaciones:

Jonathan Álvarez

- Añadir los permisos correspondientes a los scripts que los requieran para poder ejecutarse, a menudo no se entiende que el error esta relacionado al manejo de permisos en Linux
- Utilizar el modulo de Python pip para poder descargar las librerías que no se encuentren por defecto. Esta librería permite la rápida instalación de las librerías en un proceso bastante transparente

Melanny Dávila

- El uso de la correcta indentación en el desarrollo de los códigos fuente es de vital importancia dado que Python es un lenguaje que no hace uso de llaves para delimitar el funcionamiento de lazos de iteración.
- La importanción y el manejo de librerías es importante cuando se desea implementar códigos más eficientes y exactos en la obtención de datos y de esta manera se puede desarrollar aplicaciones amigables con el usuario

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] F. Doutel, "Sense Hat, Uno de los mejores periféricos para experimentar con tu Raspberry Pi: Análisis", Xataka Smart Home, feb. 29, 2016. https://www.xatakahome.com/trucos-y-bricolaje-smart/sense-hat-uno-de-los-mejores-perifericos-para-experimentar-con-tu-raspberry-pi-analisis (accedido mar. 04, 2021).

[4] "The Top 10 HATs and pHATs for Raspberry Pi". https://www.digikey.com/en/maker/blogs/2018/the-top-10-hats-and-phats-for-raspberry-pi (accedido mar. 04, 2021).
[5] "About YateBTS - YateBTS". https://wiki.yatebts.com/index.php/About_YateBTS (accedido mar. 04, 2021).
[6] "OpenAirInterface – 5G software alliance for democratising wireless innovation". https://openairinterface.org/ (accedido mar. 04, 2021).
[7] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, y C. Bonnet, "OpenAirInterface: A Flexible Platform for 5G Research", SIGCOMM Comput. Commun. Rev., vol. 44, n.º 5, pp. 33-38, oct. 2014, doi: 10.1145/2677046.2677053.
[8] "OpenAirInterface: An open platform for establishing the 5G system of the future", I'MTech, may 23, 2017. https://imtech.wp.imt.fr/en/2017/05/23/openairinterface-5g-system-of-future/ (accedido mar. 04, 2021).
[9] S. Eranol, "Instalación de dispositivos RTL-SDR en Raspberry Pi", SDR en español, abr. 10, 2020. http://sdr-es.com/2020/04/10/instalacion-rtlsdr-raspberrypi/index.html (accedido mar. 04, 2021).
[10] "YateBTS". https://wiki.yatebts.com/index.php/Main_Page (accedido mar. 04, 2021).
[11] "Building your own Portable GSM BTS at Home using The NUAND BLADERF, Raspberry PI 3 B+ and YATEBTS", Tech Belt, jun. 19, 2020. https://tech-belt.com/2020/06/19/building-your-own-portable-gsm-bts-at-home-using-the-nuand-bladerf-raspberry-pi-3-b-and-yatebts/ (accedido mar. 04, 2021).
[12] Color Theory. https://www.w3schools.com/colors/color_tryit.asp?color DarkSeaGreen (accedido mar. 04, 2021).
[13] Módulo turtle (1). Gráficos. Python. Bartolomé Sintes Marco. www.mclibre.org. https://www.mclibre.org/consultar/python/lecciones/python-turtle-1.html (accedido mar. 04, 2021).
[14] turtle — Turtle graphics — Python 3.9.2 documentation. https://docs.python.org/3/library/turtle.html (accedido mar. 04, 2021).
[15] networking - Raspberry Pi 4G, Raspberry Pi Stack Exchange. https://raspberrypi.stackexchange.com/questions/109631/raspberry-pi-4g (accedido mar. 04, 2021).