wifi_raspbian_eap

Release 1.0

Michael Guerrero

Nov 19, 2019

CONTENIDO:

1 Instalación **3**
- 1.1 Deshabilitar dhcpcd (servicio e icono en el panel o barra de herramientas) . . . 3
- 1.2 Instalar NetworkManager y las libreras necesarias para python . . . 3

2 Uso **4**
- 2.1 Impotar la libreria . . . 4
- 2.2 Conexion a redes Wifi: . . . 4

3 API **6**
- Python Module Index . . . 8
- Index . . . 9

Esta libreria es desarrollada utilizando *Python3* y la interfaz para *dbus* de *NetworkManager*, lo cual permite usar el servicio *NetworkManager*, que es un administrador de redes muy popular y robusto en sistemas Linux, a través de *Python*.

En *Raspbian* por defecto se encuentra el servicio y la interfaz grafica del administrador de conexiones *dhcpcd*, este es muy intuitivo y provee unas funcionalidades muy básicas, pero esto a su vez impide que sea posible establecer conexiones que requieran una configuración específica, como lo son las conexiones **WPA2-EAP**. Por esta razón y por la necesidad de configurar el administrador de conexiones desde *python*, se ha desarrollado esta libreria, que pone en disposición funciones de fácil uso y configuración, especialmente para conexiones:

- Sin seguridad (abiertas).
- Con protocolo de seguridad **TKIP-CCMP** (contraseña).
- Con protocolo de seguridad **EAP** (usuario y contraseña).

CHAPTER

ONE

INSTALACIÓN

1.1 Deshabilitar dhcpcd (servicio e icono en el panel o barra de herramientas)

Es necesario deshabilitar el servicio *dhcpcd* que se inicia en el proceso de arranque, para evitar tener conflictos con el servicio de *NetworkManager*, esto logra ejecutando en consola el siguiente comando:

```
sudo sytemctl disable dhcpcd
```

Además en el panel aparece un icono relacionado con la interfaz grafica de *dhcpcd*, este icono se puede deshabilitar comentando del archivo **TODO** y las siguientes lineas:

```
**TODO**
```

1.2 Instalar NetworkManager y las libreras necesarias para python

Para instalar *NetworkManger* se debe ejecutar el siguiente comando en consola:

```
sudo apt-get install network-manager
```

opcionalmente se puede instalar un inspeccionador de la interaz *dbus* con el siguiente comando:

```
sudo apt-get install d-feet
```

Se deben instalar las dependencias de python necesarias para controlar la interfaz *dbus*:

```
sudo apt install python3-gi python3-gi-cairo gir1.2-gtk-3.0
```

Y por ultimo se instala la interfaz de comunicación de *NetworkManager* para *dbus*:

```
sudo pip3 install python-networkmanager
```

CHAPTER

TWO

USO

2.1 Impotar la libreria

Para utilizar esta libreria primero debe importarla:

```
import nm_dbus_python
```

Debe asegurarse de que la carpeta nm_dbus_python se encuentra en el directorio de trabajo o en el Path.

2.2 Conexion a redes Wifi:

Para conectarse a una red Wifi sin seguridad, puede utilizar el siguiente comando:

```
nm_dbus_python.connecttoAP(ssid)
```

Para conectarse a una red Wifi con seguridad PSK, puede utilizar el siguiente comando:

```
nm_dbus_python.connecttoAP(ssid, password=pssw)
```

Para conectarse a una red Wifi con seguridad EAP, puede utilizar el siguiente comando:

```
nm_dbus_python.connecttoAP(ssid, password=pssw, user=usr)
```

en los ejemplos anteriores, los parametros ssid, pssw y usr, son cadenas de caracteres, por ejemplo:

```
nm_dbus_python.connecttoAP("wifi network", password="password", user="user")
```

ademas, la funcion connectoAP cuenta con un parametro adicional para establecer un tiempo maximo de espera para la respuesta de la conexion, por ejemplo, si se va a esperar como maximo 5 segundos a que se conecte a una red, se debe utilizar de la siguiente forma:

```
nm_dbus_python.connecttoAP("wifi network", password="password", timeout=5)
```

si se conoce el bssid (direccion MAC) de la red se puede utilizar el siguiente comando:

```
nm_dbus_python.connecttoAP("",bssid="CC:35:40:98:A8:3F", password="password")
```

tambien se puede ingresar el bssid y el ssid de la red:

```
nm_dbus_python.connecttoAP("wifi network",bssid="CC:35:40:98:A8:3F", password=
↪"password")
```

en este caso se tratara de conectar a la red “wifi network”, en caso de encontrar la red o de no poder acceder a ella, intentara conectarse a la red con bssid = "CC:35:40:98:A8:3F"

CHAPTER

THREE

API

nm_dbus_python.**accessPoints**(*prop*, *value*)
Function used to get available access points.

Parameters

- **prop** (*str*) – is the property that must have an access point of interest
- **value** (*str*) – is the respective value that property must have

Returns A tuple with availables access points and the wireless device

Return type tuple

nm_dbus_python.**connectProfile**(*setting*, *key*, *value*)
Function used to get previously saved connection profiles.

Parameters

- **setting** (*str*) – is the setting that must have a connection profile
- **key** (*str*) – is the respective key that a property of setting must have
- **value** (*str*) – is the respective value of key must have

Returns A settings object

nm_dbus_python.**deactive_wireless_conn**()
Function used to disconnect all active wireless connections.

nm_dbus_python.**get_all_ap_info**()
Function used to get the attributes of interest of the available networks.

Returns a list of dictionaries with the attributes in attr_interest list

Return type list

nm_dbus_python.**create_connProfile**(*ssid*, *bssid=''*, *user=''*, *password=''*)
Function used to create a new connection profile.

Parameters **ssid** (*str*) – is the ssid of the access point.

Returns A dictionary with all the information necessary to create a connection profile

Return type dict

nm_dbus_python.**connecttoAP**(*ssid*, *bssid=''*, *user=''*, *password=''*, *timeout=-1*)
This functions try to connect to a network with the given parameters

Currently this function can only connecto to:

- open networks

- psk tkip networks
- peap mschapv2 networks

this function prioritizes ssid over bssid always return the bssid of the network connected to

Parameters

- **ssid** (*str*) – is the ssid of the access point to connect to
- **timeout** (*int*) – is the numbers of seconds to wait for connection, -1 for wait until connect

Returns False if the connection failed or a string with the MacAddress

Return type bool|str

nm_dbus_python.**connectMAC**(*bssid*, *user=''*, *password=''*)

This functions try to connect to a network with the given bssid(MacAddress)

Currently this function can only connecto to:

- open networks
- psk tkip networks
- peap mschapv2 networks

this function prioritizes ssid over bssid always return the bssid of the network connected to

Parameters **bssid** (*str*) – is the bssid(MacAddress) of the access point to connect to

Returns False if the connection failed or a string with the MacAddress

Return type bool|str

nm_dbus_python.**JaverianaCali**(*user*, *password*)

Function used to connect to the network JaverianaCali with the given user and password.

Is assumed that JaverianaCali is an 802.1x connection. This function is only for test, use connecttoAP instead.

Parameters

- **user** (*str*) – is a string with a valid user
- **password** (*str*) – is a string with the password of the user

nm_dbus_python.**Invitados_JaverianaCali**()

Function used to connect to the network 'Invitados JaverianaCali'.

Is assumed that Invitados JaverianaCali is open. This function is only for test, use connecttoAP instead.

Parameters

- **user** (*str*) – is a string with a valid user
- **password** (*str*) – is a string with the password of the user

PYTHON MODULE INDEX

n

nm_dbus_python, 6

INDEX

A

accessPoints() (*in module nm_dbus_python*), 6

C

connectMAC() (*in module nm_dbus_python*), 7
connectProfile() (*in module nm_dbus_python*), 6
connecttoAP() (*in module nm_dbus_python*), 6
create_connProfile() (*in module nm_dbus_python*), 6

D

deactive_wireless_conn() (*in module nm_dbus_python*), 6

G

get_all_ap_info() (*in module nm_dbus_python*), 6

I

Invitados_JaverianaCali() (*in module nm_dbus_python*), 7

J

JaverianaCali() (*in module nm_dbus_python*), 7

N

nm_dbus_python (*module*), 6