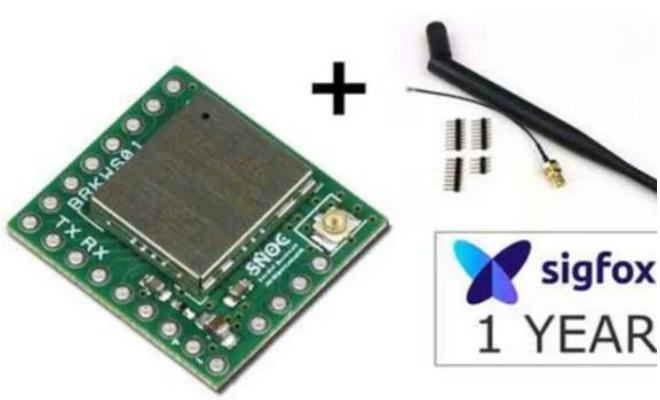
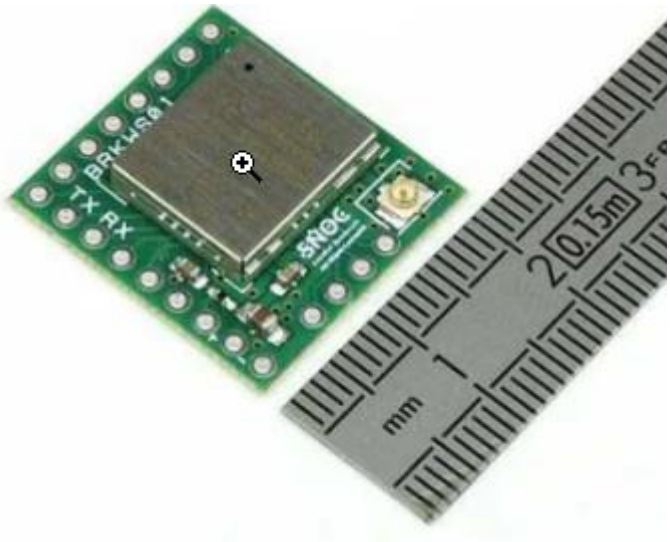
SigFox Python et Raspberry Pi

I. Matériel

1. Présentation de la carte



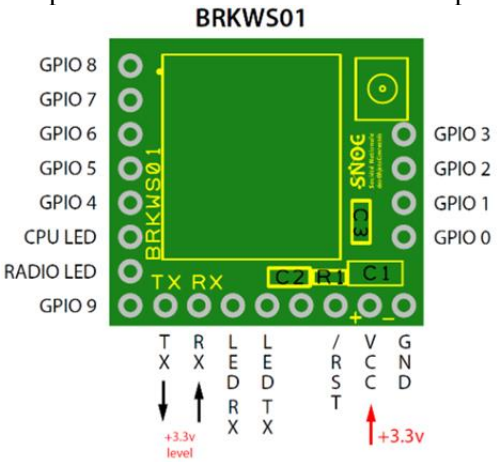


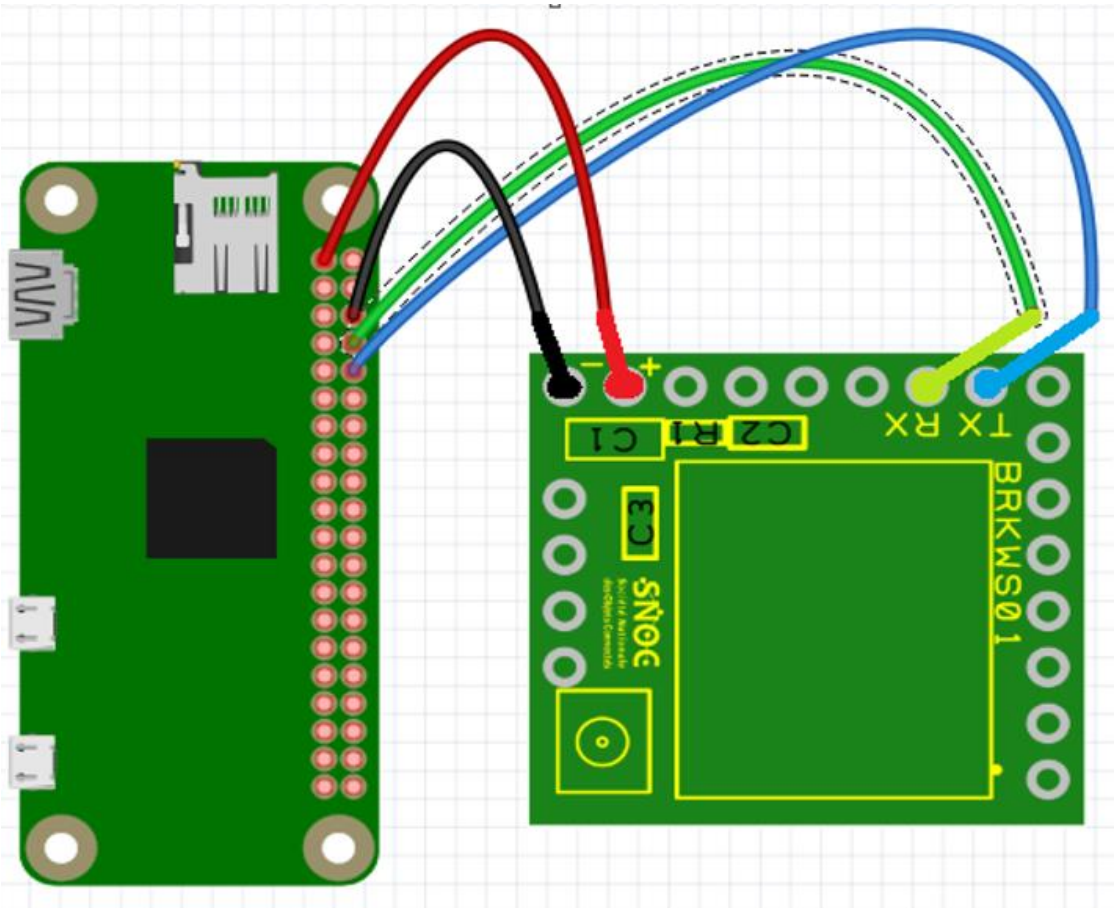
Pour faire simple, c'est un module de communication pour le réseau sigfox qui fonctionne à l'aide de commande AT. Cette carte, vendue par snoc, intégre un module du fabricant coréen Wisol :

A noter que la carte complète avec l'antenne coûte environ 30 € et le module seul 10 fois moins.

2. Câblage

Simplement sur la liaison série de la Rpi et l'alimentation en 3.3V :





3. Documentations

a. Carte snooc :

https://docs.rs-online.com/bf2a/0900766b815bf8c5.pdf et le PCB https://docs.rs-online.com/78d2/0900766b815bf8c2.pdf

b. Commande AT

https://docs.rs-online.com/51f6/0900766b815bf8c3.pdf

4. Pour plus d'explications

https://www.framboise314.fr/carte-de-prototypage-sigfox-par-snoc/

II. Le code

Malheureusement le code fourni par snooc ne fonctionne pas sur les dernières versions de raspbian, probablement à cause de la gestion du port série.

1. Coder en octets

Le problème est qu'il faille envoyer uniquement des octets :

```
Running: eez.py
Traceback (most recent call last):
  File "/home/pi/mu_code/eez.py", line 85, in <module>
    ser.write('bAT$SF=')
  File "/usr/lib/python3/dist-packages/serial/serialposix.py", line 532, in write
    d = to_bytes(data)
  File "/usr/lib/python3/dist-packages/serial/serialutil.py", line 63, in to_bytes
    raise TypeError('unicode strings are not supported, please encode to bytes: {!r}'.format(seq))
TypeError: unicode strings are not supported, please encode to bytes: 'bAT$SF='
>>>
```

`str.encode()` renvoie une représentation `bytes` de la chaîne Unicode, codée dans l'encodage *encoding* demandé. (https://docs.python.org/fr/3/howto/unicode.html)

J'ai donc créé une fonction qui envoie la donnée en octets sur le port série :

```
def EnvoyerEnOctet(commande): #Conversion en octets
    ser.write(commande.encode())
```

2. Sigfox

a. Format de la trame : tout en majuscule

La commande AT pour envoyer doit commencer par AT$SF et doit se terminer par \r. Par exemple :

```
EnvoyerEnOctet("AT$SF=AABBCC\r") #il faut que ce soit des majuscules
```

Attention : il faut que la trame soit en majuscule.
Par exemple :

```
EnvoyerEnOctet("AT$SF=aaBBCC\r") # Ne fonctionne pas
```

b. Commandes utiles

La commande « AT » permet de vérifier la communication

```
EnvoyerEnOctet("AT\r\n")# Permet de vérifier la communication
data=ser.read(2)         #La réponse doit être OK
print(data)
```

La commande « AT$I=10» permet de connaître l'identifiant (unique) du module de communication sigfox qui se trouve sur la carte électronique

```
print("Get ID")
EnvoyerEnOctet("AT$I=10\r\n") # Demande de l'ID de l'appareil
data=ser.read(10)
```

La commande « AT$I=11» permet de connaître le numéro PAC

```
print("Get PAC")
EnvoyerEnOctet("AT$I=11\r\n") # Demande du numéro PAC
data=ser.read(18)
print(data)
```

La fonction qui regroupe ces commandes est la suivante :

```
def SigfoxInfo():
    EnvoyerEnOctet("AT\r\n")# Permet de vérifier la communication
    data=ser.read(2)         #La réponse doit être OK
    print(data)
    print("Get ID")
    EnvoyerEnOctet("AT$I=10\r\n") # Demande de l'ID de l'appareil
    data=ser.read(10)
    print(data)
    print("Get PAC")
    EnvoyerEnOctet("AT$I=11\r\n") # Demande du numéro PAC
    data=ser.read(18)
    print(data)
```

c. Des nombres hexadécimaux mais envoyés en caractères.

Reprenons l'exemple :

```
EnvoyerEnOctet("AT$SF=AABBCC\r") # valeur de la trame se trouve après AT$SF
```

Cette commande envoie la trame AABBCC :

- C'est une chaîne de caractères
- Mais elle représente la valeur 0xAA 0xBB 0xCC

Ainsi

```
EnvoyerEnOctet("AT$SF=HHBBCC\r") #Erreur !!!
```

Ne fonctionne pas puisque le nombre hexadécimal 0xHH n'existe pas.

d. Envoyer des valeurs décimales dans le bon format

Pour envoyer des valeurs décimales il faut :

1. Vérifier que la valeur soit représentée sur un seul octet, donc la valeur <=255
2. Convertir en hexadécimal
3. Convertir ce nombre en chaîne de caractères
4. Supprimer les caractères « 0x » devant ce nombre
5. Ajouter un « 0 » devant le nombre si celui-ci est représenté sur un seul caractère par exemple la conversion de 10 donne 0xA or il faut envoyer 0A

Voici la fonction qui effectue ces tâches :

```
def ConversionDeciHexStr(valeur) :
# converti un nombre <256 en chaine de caractères hexa sans les '0x' devant
    if valeur >255: #sur un octet maxi
        return ''
    valHexStr = str(hex(valeur)) # converti en hexa puis en String
    valMajuscule = valHexStr.upper() # tout en majuscule
    valDeuxDerniersCaracteres = valMajuscule[2:]
    if len(valDeuxDerniersCaracteres) == 1 : #conversion sur un seul hexa
        valDeuxDerniersCaracteres = '0'+valDeuxDerniersCaracteres #ajout d'un zéro devant
    return valDeuxDerniersCaracteres
```

III. Code complet

```
from time import sleep
import serial


def EnvoyerEnOctet(commande): #envoi en octet
    ser.write(commande.encode())

def SigfoxInfo():
    EnvoyerEnOctet("AT\r\n")# Permet de vérifier la communication
    data=ser.read(2)         #La réponse doit être OK
    print(data)
    print("Get ID")
    EnvoyerEnOctet("AT$I=10\r\n") # Demande de l'ID de l'appareil
    data=ser.read(10)
    print(data)
    print("Get PAC")
    EnvoyerEnOctet("AT$I=11\r\n") # Demande du numéro PAC
    data=ser.read(18)
    print(data)

def ConversionDeciHexStr(valeur) :# converti un nombre <256 en chaine de caractères hexa s
ans les '0x' devant
    if valeur >255: #sur un octet maxi
        return ''
    valHexStr = str(hex(valeur)) # converti en hexa puis en String
    valMajuscule = valHexStr.upper() # tout en majuscule
    valDeuxDerniersCaracteres = valMajuscule[2:]
    if len(valDeuxDerniersCaracteres) == 1 : #conversion sur un seul hexa
        valDeuxDerniersCaracteres = '0'+valDeuxDerniersCaracteres #ajout d'un zéro devant
    return valDeuxDerniersCaracteres

def SigfoxSend():
    #Initiate a Transmission
    print('AT')
    EnvoyerEnOctet("AT\r\n")
    data=ser.read(4)
    print(data)
    print("Init Transmission")
    sleep(1)

    #il faut que ce soit des majuscules Fonctionne ok
    #EnvoyerEnOctet("AT$SF="+ConversionDeciHexStr(123)+"AABBCC\r")
    EnvoyerEnOctet("AT$SF="+ConversionDeciHexStr(123)+ConversionDeciHexStr(14)+"\r") #idem
 AT$SF=7B0E\t
    #EnvoyerEnOctet("AT$SF=AABBCC\r")
```

```
    sleep(6)
    data=ser.read(4)        # Réponse OK mais il peut y avoir des perts
    print(data)


# Initialise le port série
ser=serial.Serial("/dev/ttyS0")
ser.baudrate=9600
##----------

if ser.isOpen:  # Vérification port ouvert
    print("Le Port est ouvert")
    #SigfoxInfo()
    SigfoxSend()
else:
    print("Impossible d'ouvrir le port")
ser.close()
```