27/03/2017

Manuel d'utilisation

Decodeb64.py, V1.1

python

Etienne BOURDEAU
LABORATOIRE DE PHYSIQUE CORPUSCULAIRE, AUBIERE

Table des matières

Introduction au programme

Le programme « Decodeb64.py » est un projet initié par M. ROYER Laurent, du Laboratoire de Physique Corpusculaire d'Aubière, puis achevé par M. BOURDEAU Etienne, du même laboratoire.

Initialement réalisé sous Matlab, il a été repris sous Python 3.6.

Ce programme sert à analyser les paquets capturés par la passerelle « Multitech Multiconnect Conduit (MTCDT) », décoder les coordonnées GPS encodées en base64 et les traduire, récupérer les informations utiles au laboratoire (GPS, Puissance du Signal, Rapport Signal sur Bruit et date / heure) sur un fichier à part.

Analyse du programme

Le programme « Decodeb64.py » a été conçu pour analyser un fichier de la forme :


```
lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":2,"codr":"4/5","data":"ACdaFa2z9TBg3McAAAAAgACeWm82FDU=","datr":"SF12BW125","freq":868.5,"lsnr":8,"modu":"LORA","rfc
":16,"timestamp":"2017-03-06T19:32:24.589692Z","tmst":591043820}lora/00-80-00-00-00-00-c7-dc/packet_sent {"codr":"4/5","data":"YAEAAAYgAABmOOij","datr":"SF7BW125","f
lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":3,"codr":"4/5","data":"gAEAAAYAAgABRjLn5H21UunhWcId3LEs","datr":"SF7BW125","freq":867.10000000000002,"lsnr":9.800000
2:56.147857Z","tmst":622596803}lora/00-80-00-00-00-00-c7-dc/up {"chan":1,"cls":0,"codr":"4/5","data":"ABcAQRVZQwI2E04=","datr":"SF7BW125","freq":"868.3","lsnr":"10",
ssi":-84,"seqn":4,"size":16,"timestamp":"2017-03-06T19:33:06.183094Z","tmst":632633659}lora/00-80-00-00-00-00-c7-dc/packet_sent {"codr":"4/5","data":"YAEAAAYgBACqyQ8
A","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":643670444}lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":0,"codr":"4/5","data":"gAEAAAYABgABKmhdmmADibkImSPkpQV
RA","rfch":1,"rssi":-93,"size":24,"stat":1,"time":"2017-03-06T19:33:36.299821Z","tmst":662744028}lora/00-80-00-00-00-00-c7-dc/up {"chan":5,"cls":0,"codr":"4/5","data
du":"LORA","opts":"","port":1,"rfch":1,"rssi":-87,"seqn":8,"size":16,"timestamp":"2017-03-06T19:33:46.336449Z","tmst":672780828}lora/00-80-00-00-00-00-c7-dc/packet_s
rue,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":683817571}lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":4,"codr":"4/5","data":"gAEAAAYACgABr7fo
ORA","rfch":0,"rssi":-95,"size":24,"stat":1,"time":"2017-03-06T19:34:16.438353Z","tmst":702891139}lora/00-80-00-00-00-00-c7-dc/up {"chan":0,"cls":0,"codr":"4/5","dat
0006000c00","modu":"LORA","opts":"","port":1,"rfch":0,"rssi":-91,"seqn":12,"size":16,"timestamp":"2017-03-06T19:34:26.482989Z","tmst":712927923}lora/00-80-00-00-00-0
F7BW125","freq":867.70000000000005,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":723964756}lora/00-80-00-00-00-00-c7-dc/packet_recv {"ch
"SF7BW125","freq":867.5,"lsnr":7.5,"modu":"LORA","rfch":1,"rssi":-64,"size":24,"stat":1,"time":"2017-03-06T19:34:56.588739Z","tmst":743038276}lora/00-80-00-00-00-00-
","mhdr":"80010000060010000","modu":"LORA","opts":"","port":1,"rfch":1,"rssi":-70,"seqn":16,"size":16,"timestamp":"2017-03-06T19:35:06.621329Z","tmst":753075116}lora/
freq":868.5,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":764111883}lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":4,"codr":"4/5","dat
r":10,"modu":"LORA","rfch":0,"rssi":-87,"size":24,"stat":1,"time":"2017-03-06T19:35:36.740230Z","tmst":783185411}lora/00-80-00-00-00-00-c7-dc/up {"chan":1,"cls":0,"c
006001400","modu":"LORA","opts":"","port":1,"rfch":0,"rssi":-91,"seqn":20,"size":16,"timestamp":"2017-03-06T19:35:46.769579Z","tmst":793222187}lora/00-80-00-00-00-00
998,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":804258964}lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":0,"codr":"4/5","data":"gAEA
.5,"modu":"LORA","rfch":1,"rssi":-97,"size":24,"stat":1,"time":"2017-03-06T19:36:16.882603Z","tmst":823332548}lora/00-80-00-00-00-00-c7-dc/up {"chan":5,"cls":0,"codr
00006001800","modu":"LORA","opts":"","port":1,"rfch":1,"rssi":-106,"seqn":24,"size":16,"timestamp":"2017-03-06T19:36:26.914226Z","tmst":833369420}lora/00-80-00-00-00
"freq":867.10000000000002,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":844406195}lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":4,"co
9999999999995,"lsnr":7.7999999999999998,"modu":"LORA","rfch":0,"rssi":-100,"size":24,"stat":1,"time":"2017-03-06T19:36:57.029182Z","tmst":863479683}lora/00-80-00-00-
":868.5,"lsnr":5.5,"mhdr":"8001000006001c00","modu":"LORA","opts":"","port":1,"rfch":0,"rssi":-103,"seqn":28,"size":16,"timestamp":"2017-03-06T19:37:07.063456Z",
4","datr":"SF7BW125","freq":867.70000000000005,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":884553188}lora/00-80-00-00-00-00-c7-dc/pack
B53Pzb+","datr":"SF7BW125","freq":868.5,"lsnr":4,"modu":"LORA","rfch":0,"rssi":-107,"size":24,"stat":1,"time":"2017-03-06T19:38:08.994957Z","tmst":93545243}lora/00-
","freq":"867.9","lsnr":"6.5","mhdr":"8001000006002200","modu":"LORA","opts":"","port":1,"rfch":1,"rssi":-107,"seqn":34,"size":16,"timestamp":"2017-03-06T19:38:19.03
data":"YAEAAAYgIQAVWxnb","datr":"SF7BW125","freq":867.29999999999995,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":956578675}lora/00-80-
","datr":"SF7BW125","freq":867.29999999999995,"lsnr":8,"modu":"LORA","rfch":1,"rssi":-97,"size":24,"stat":1,"time":"2017-03-06T19:38:49.253819Z","tmst":975705435}lor
"datr":"SF7BW125","freq":"868.1","lsnr":"9.5","mhdr":"8001000006002600","modu":"LORA","opts":"","port":1,"rfch":0,"rssi":-99,"seqn":38,"size":16,"timestamp":"2017-03
acket_sent {"codr":"4/5","data":"YAEAAAYgJQB60Mei","datr":"SF7BW125","freq":868.29999999999995,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"t
ata":"gAEAAAYAKQABLExPa6jvai7HajULTnGR","datr":"SF7BW125","freq":868.10000000000002,"lsnr":6.5,"modu":"LORA","rfch":0,"rssi":-103,"size":24,"stat":1,"time":"2017-03-
cls":0,"codr":"4/5","data":"ABEAQRUWxwI2H2w=","datr":"SF7BW125","freq":867.3","lsnr":"9","mhdr":"8001000006002a00","modu":"LORA","opts":"","port":1,"rfch":1,"rssi":
-00-00-00-00-c7-dc/packet_sent {"codr":"4/5","data":"YAEAAAYgKQAUHsvE","datr":"SF7BW125","freq":867.5,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size
:"4/5","data":"gAEAAAYALQABO7ejbyuyub0xvvSUAST1","datr":"SF7BW125","freq":868.10000000000002,"lsnr":10.5,"modu":"LORA","rfch":0,"rssi":-87,"size":24,"stat":1,"time":
"chan":1,"cls":0,"codr":"4/5","data":"ABEAQRT4LAI2QcE=","datr":"SF7BW125","freq":"868.3","lsnr":"10","mhdr":"8001000006002e00","modu":"LORA","opts":"","port":1,"rfch
":1076220924}lora/00-80-00-00-00-00-c7-dc/packet_sent {"codr":"4/5","data":"YAEAAAYgLQDdGcY4","datr":"SF7BW125","freq":867.89999999999998,"ipol":true,"modu":"LORA","
00-80-00-00-00-00-c7-dc/packet_recv {"chan":6,"codr":"4/5","data":"gAEAAAYAMQABI/lmR9hAkY6/oN13lWCI","datr":"SF7BW125","freq":867.70000000000005,"lsnr":10.1999999999
9:41:00.182401Z","tmst":1106630307}lora/00-80-00-00-00-00-c7-dc/up {"chan":4,"cls":0,"codr":"4/5","data":"ABEAQRUOsWI2UaO=","datr":"SF7BW125","freq":"867.3","lsnr":"
"rfch":0,"rssi":-22,"seqn":51,"size":16,"timestamp":"2017-03-06T19:41:10.365868Z","tmst":1116816083}lora/00-80-00-00-00-00-c7-dc/packet_sent {"codr":"4/5","data":"YA
A","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":1127852788}lora/00-80-00-00-00-00-c7-dc/packet_recv {"chan":0,"codr":"4/5","data":"gAEAAAYANQABUPuBZSWJdIpVFejllx
":"LORA","rfch":1,"rssi":-21,"size":24,"stat":1,"time":"2017-03-06T19:41:40.468883Z","tmst":1146926284}lora/00-80-00-00-00-00-c7-dc/up {"chan":5,"cls":0,"codr":"4/5"
3700","modu":"LORA","opts":"","port":1,"rfch":0,"rssi":-23,"seqn":55,"size":16,"timestamp":"2017-03-06T19:41:50.656236Z","tmst":1157112179}lora/00-80-00-00-00-00-c7-
999995,"ipol":true,"modu":"LORA","ncrc":false,"powe":11,"rfch":0,"size":12,"tmst":1168148891}lora/00-80-00-00-00-00-c7-dc/packet_recv
```


Figure 1:Fichier d'entrée "Data_6mars_cezeaux.txt"

I. Importation des modules :

Python 3.6 possède un grand nombre de librairies pratiques :

```
import base64
import os
```

➔ Import du module base64, qui permet d'encoder, et de décoder, la Base64

➔ Import du module os, qui permet d'interagir avec les fichiers

II. Mise en forme du fichier d'entrée / Création du fichier intermédiaire

Sous la forme de paquet, il est difficile de lire / comprendre / analyser les informations. Le programme permet donc de générer un fichier intermédiaire, contenant les mêmes informations, mises en forme. Ce fichier est automatiquement détruit par le programme après que les informations voulues aient été extraites.

La signification des termes présents dans ce fichier se trouve dans le glossaire.

```
with open('./Tests/data_6mars_cezeaux.txt', 'r') as f:
    with open('./fichier_intermédiaire.txt', 'w') as f2:
        filedata = f.read()
        filedata = filedata.replace(',', '\n').replace(' ', '\n').replace('"', '').replace('{', '').replace('}', '\n')\
            .replace('(null)', '(null) \n')
        f2.write(filedata)
```

1 – On ouvre notre fichier d'entrée en lecture, qu'on surnomme « f »

2 – On ouvre un fichier intermédiaire en écriture, qu'on surnomme « f2 »

3 – On crée la variable « filedata », qui lit l'intégralité du fichier d'entrée.

4- A partir du fichier lu, la variable met en forme son propre contenu : les « , » et « } » sont remplacées par des passages à la ligne (« \n » en code), les espaces et « { » sont annulés (remplacés par une chaine vide), et les « (null) » (qu'on trouve dans les paquets vides), sont remplacées par « (null) \n », afin de conserver l'information null et d'y ajouter un passage à la ligne pour annoncer le paquet suivant.

5 – On écrit le contenu de la variable filedata (soit le contenu mis en forme du fichier d'entrée) dans le fichier intermédiaire. Ce fichier sera écrasé à chaque nouvelle exécution du programme.

III. Analyse du fichier intermédiaire / Décodage de la base64

Cette partie est le cœur du programme, et devait initialement représenter sa seule fonctionnalité.

```
with open(mid_file, 'r') as f:
    with open('./out_file.txt', 'w') as f2:
            for line in f:
                if 'lora/' and 'up' in line:
                    for line in f:
                        if 'data' in line:  # Base 64 decoding loop
                            souschaine = line[5:len(line)]
                            decodage = base64.b64decode(souschaine)
                            liste = list(decodage)
                            temp = liste[1]
                            latitude = (((liste[6] + (liste[5] * 256) + (liste[4] * 256 * 256) + (liste[3] * 256 * 256 *
                                                                    256)) / ((2 ** 31) - 1) * 90))
                            lat2 = round(latitude, 4)

                            longitude = ((liste[10] + (liste[9] * 256) + (liste[8] * 256 * 256) + (liste[7] * 256 * 256
                                                                    * 256)) / ((2 ** 31) - 1) * 180)
                            lng2 = round(longitude, 4)
                            resultat = ('température = ', (str(temp)), '°C', '\n', 'latitude = ', (str(lat2)),
                                        '°', '\n', 'longitude = ', (str(lng2)), '°', '\n')
                            result2 = ''.join(resultat)
                            f2.write(result2)
```

1 – On ouvre le fichier intermédiaire précédemment créé en lecture, surnommé f.

2 – On ouvre le fichier final (ici nommé fichier de sortie) en écriture, surnommé f2. Ce fichier sera écrasé à chaque exécution du programme.

3 – Pour simplifier la lecture, je vais recopier le code et sa signification (à droite) :

```
For line in f :                                          Pour chaque ligne du fichier intermédiaire
  If 'lora/' and 'up' in line :                          Si on trouve une ligne contenant lora/ et up
    For line in f:                                       Pour chaque ligne du fichier intermédiaire (à partir du motif trouvé)
      If 'data' in line :                                Si on trouve une ligne contenant 'data '
        Souschaine = line[5:len(line)]                   On crée une variable souschaine, qui lit du 6e caractère* à la fin de la ligne
        Decodage = base64.b64decode(souschaine)          Grâce au module base64, on décode la souschaine**
        Liste = list(decodage)                           On transforme les octets décodés en liste, pour pouvoir les multiplier
        Temp = liste[1]                                  On récupère la température, située dans le 2e élément de la liste, dans une variable

        latitude = (((liste[6] + (liste[5] * 256) + (liste[4] * 256 * 256) + (liste[3] * 256 * 256 * 256)) / ((2 ** 31) - 1) * 90))
        lat2 = round(latitude, 4)

                                                         Calcul de la latitude : trouvé sur le site du constructeur, lien disponible dans les sources, arrondi à 4 chiffres

        longitude = ((liste[10] + (liste[9] * 256) + (liste[8] * 256 * 256) + (liste[7] * 256 * 256 * 256)) / ((2 ** 31) - 1) * 180)
        long2 = round(longitude, 4)
                                                         Calcul de la longitude : idem que latitude

        resultat = ('température = ', (str(temp)), '°C', '\n', 'latitude = ', (str(lat2)), '°', '\n', 'longitude = ', (str(long2)), '°', '\n')

                                                         Mise en forme des informations décodées afin qu'elles soient identifiables par l'utilisateur.

        result2 = ''.join(resultat)                      Join permet de transformer Resultat (tuple) en chaine de caractères
        f2.write(result2)                                On écrit le résultat mis en forme dans le fichier de sortie
```

* En Python, les compteurs commencent à 0, ce qui fait que pour sélectionner le 6e caractère, il faut écrire 5.

** La sous-chaine contient uniquement la partie codée en base64. Elle est décodée dans sa forme d'origine, ici en octets

IV. Récupération des autres données

Les autres données n'étant pas codées, elles sont beaucoup plus simples à réunir et écrire dans le fichier de sortie. Les procédés utilisés sont exactement les mêmes que pour le décodage des coordonnées, depuis la partie « Mise en forme des informations décodées […] à la fin.

Le programme itère sur chaque ligne lue et vérifie la présence de chaque motif grâce à l'opérateur (elif) (traduit opérateur (et si) :

```
elif 'lsnr:' in line:
    lsnr = ('Rapport Signal / Bruit = ', line[5:len(line)])
    lsnr2 = ''.join(lsnr)
    # print('lsnr = ', lsnr)
    f2.write(lsnr2)
elif 'rssi:' in line:
    rssi = ('Puissance du signal = ', line[5:len(line)])
    rssi2 = ''.join(rssi)
    # print('rssi = ', rssi)
    f2.write(rssi2)
elif 'timestamp:' in line:
    année = line[10:14]
    mois = line[15:17]
    jour = line[18:20]
    heure = line[21:23]
    minutes = line[24:26]
    secondes = line[27:29]
    time = (jour, '-', mois, '-', année, ', à ', heure, 'h', minutes, ' (', secondes,
            ' secondes)', '\n', '\n')
    time2 = ''.join(time)
    # print('time = ', time)
    f2.write(time2)
if 'tmst:' in line:
    break
```

Mise en forme des données temporelles

La boucle finit si 'tmst' est rencontré, car chaque paquet finit par ce motif.

V. Fin du programme

```
os.remove(mid_file)  # Comment for DEBUG
```

1- Grâce au module os, on supprime le fichier intermédiaire. On peut commenter cette ligne (grâce au caractère #) afin de le conserver pour débug ou vérifier les informations traitées par la boucle.

Annexes

Coder et décoder en base64 manuellement.

Afin de vérifier la véracité des informations décodées par le programme, on peut vouloir décoder manuellement un code en base 64. Voici la méthode pour le faire.

Coder en base64

Pour comprendre comment décoder, il faut comprendre comment coder en base64. Ici, notre objectif va être de coder le caractère « a ».

*1 – Convertir le caractère en ASCII (table ASCII disponible en annexe). En décimale, en ASCII, a vaut **97**.*

a = 97

2 – Convertir le résultat en binaire :

97 = 0110 0001
(soit $1*2^0 + 0*2^1 + 0*2^2 + 0*2^3 + 0*2^4 + 1*2^5 + 1*2^6 + 0*2^7$) = 1 + 32 + 64

3 – Séparer l'octet obtenu en groupes de 6. Si le dernier groupe n'est pas complet, rajoutez autant de 0 que nécessaire à droite ce dernier jusqu'à obtenir un groupe complet.

01100001 = 011000 01(0000)

4 – Convertir les groupes en binaire en décimale, séparément :

011000 = 24
010000 = 16

5 – On remplace les chiffres obtenus par leur équivalent dans l'alphabet de la base64 (Disponible en annexe):

24 = Y
16 = Q

6 – Un code en base64 se compose toujours d'un groupe de 4 caractères. Ici, nous n'avons que Y et Q. Il est donc possible de rajouter jusqu'à deux signes « = » pour obtenir un groupe complet. Le caractère « = » n'a aucune valeur.

a = YQ==

Décoder la base64

Nous pouvons à présent plus facilement appréhender le décodage de la base64.
Pour ce faire, nous allons prendre un code facile : QnJhdm8=

1 – On sépare le code en groupes de 4 :

QnJh dm8=

2 – On prend le premier groupe : QnJh, et on le convertit en groupes de 6 en binaire (selon l'alphabet base64) :

Q = 010000
n = 100111
J = 001001
h = 100001

3 – On regroupe ces groupes en octets :

010000[10] 0111[0010] 01[100001]

4 – On convertit chaque octet en décimale :

01000010 : 2 + 64 = 66
01110010 : 2 + 16 + 32 + 64 = 114
01100001 : 1 + 32 + 64 = 97

Converti via la table ASCII, on obtient : **B r a**

5 – Idem pour le second groupe, selon le même exemple :

- dm8= 011101 100110 111100 = (Rappel : = n'a pas de valeur)
- mise en octets : 01110110 01101111 00000000
- conversion en décimale : 118 111 0
- Conversion décimale -> ASCII : **v o**

6 – En mettant les résultats bout à bout, on obtient **B r a v o**.
Le décodage est complet.

Contenu des paquets LoRa envoyés par un Mdot

Les messages MQTT (MQ Telemetry Transport, soit les paquets reçus par la passerelle Multitech via le client Mosquitto) prennent toujours la même forme, et le programme Decodeb64.py ne s'intéresse qu'aux parties « dc /up » de ces messages.

Il est toutefois intéressant de savoir de quelles parties se composent les échanges entre les capteurs Mdot et la passerelle Multiconnect Conduit :

Le contenu précis de chaque message MQTT est disponible sur le site du constructeur (lien dans les sources)

1 – **Lora / Joined** : Jonction du nœud (la passerelle)

2 – **Lora / up** : Réception de paquet montant ; les informations redondantes sont supprimées, et le paquet est authentifié par la passerelle.

3 – **Lora / packet_recv** : les informations redondantes ne sont pas supprimées, le paquet n'est pas authentifié par la passerelle.

4 – **Lora / packet_sent** : Un paquet a été envoyé depuis la passerelle vers le capteur.

5 – Il existe un 5^{e} type de message, le **Lora / Queue full**, qu'à l'heure actuelle nous n'avons pas rencontré. Il semble désigner que le nombre de messages maximum recevables simultanément par un seul nœud a été atteint.

Termes propres à Python :

Comme chaque langage, Python possède sa propre syntaxe. Cette partie va vous aider à vous repérer si vous n'avez pas de connaissances particulières dans ce langage :

```
with open('Chemin/du/fichier.txt', 'r') as f:
```

Permet d'ouvrir un fichier texte, et de le refermer automatiquement après la fin de la fonction / de la boucle, évitant de terminer par un fichier.close().

Il existe plusieurs modes d'ouverture, mais les plus utilisés sont :

- r, pour l'ouverture en lecture.
- w, pour l'ouverture en écriture.
- a, pour l'ouverture en ajout.

La partie 'as f :' permet de créer un alias pour le fichier ouvert, ce qui permet de l'appeler ultérieurement plus facilement. Par exemple, il sera possible d'écrire ***f.read()***, plutôt que ***('./Chemin/du/fichier.txt').read().***

```
filedata = f.read()
```

f.read(), justement. D'abord, ici, on initialise une variable, nommée filedata, qui permet de lire le fichier précémment ouvert. Sous Python, il existe 3 modes de lecture :

.read(), qui permet de lire tout le fichier d'un coup
.readline(), qui ne lit que la première ligne du fichier
.readlines(), qui lit le fichier ligne à ligne et inscrit chaque ligne dans une liste

```
filedata.replace('caractère_à_remplacer', ''caractère_de_remplacement')
```

Ici, on appelle notre variable filedata (qui lit toujours notre fichier), et on remplace un caractère par un autre. On peut, à la place de caractère_de_remplacement, laisser une chaine vide '' si on veut simplement supprimer le caractère. On peut aussi utiliser '\n' comme caractère de remplacement, pour effectuer un passage à la ligne.

```
for line in f:
```

Dans Python 3.6, le terme « line » n'a pas besoin d'être défini, il fait partie des mots-clés du langage. Il désigne automatiquement une ligne d'un fichier. « For line in f » désigne donc automatiquement « pour chaque ligne dans f »

```
souschaine = line[début de la souschaine(chiffre uniquement):len(line)]
```

On initialise une variable souschaine afin d'extraire uniquement une partie de la ligne qui nous intéresse en vue de décodage. Il faut donc lui indiquer que la valeur de cette souschaine va du caractère qui nous intéresse dans la ligne (ici line[5, puisqu'on veut partir du 6^{e} caractère) jusqu'à la fin de la ligne(:len(line), len définissant length, la longueur totale de la ligne)

```
result2 = ''.join(resultat)
```

En Python 3.6, seules les chaines de caractère sont inscriptibles dans un fichier. Dans decodeb64.py, les résultats sont mis en forme avec plusieurs chaines de caractères, ce qui forme un « tuple », un tableau. La fonction ''.join permet de transformer ce tableau en chaine, afin de l'écrire dans notre out_file.txt.

Sources

Base64

Comprendre le décodage de la base64 (Anglais) :
http://forums.devshed.com/python-programming/958996-trying-understand-decode-base64-post2927595.html

Opérations à effectuer pour obtenir la latitude et la longitude (Anglais) :
http://www.multitech.net/developer/software/dot-box-and-evb-software/survey-gps/

LoRaWAN packet decoder (Anglais) :
https://runkit.io/avbentem/lorawan-packet-decoder/branches/master?data=ACdaFa2z9TBg3McAAAAAgACeWm82FDU%3D

Table ASCII :
https://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange

Alphabet Base64 :
https://fr.wikipedia.org/wiki/Base64

Informations relatives aux messages MQTT

Contenu des messages MQTT (Anglais)
http://www.multitech.net/developer/software/lora/lora-network-server/mqtt-messages/