27/03/2017

Datasheet

Decodeb64.py, V1.1

Etienne BOURDEAU
LABORATOIRE DE PHYSIQUE CORPUSCULAIRE, AUBIERE

Summary

Introduction to the program 2

Program Analyzing 2

I. Modules import 2

II. Laying out of in_file / Creation of mid_file 3

III. Mid_File Analyze / Base64decode 3

IV. Other data recovery 5

V. End of the program 5

Appendix 6

Manually code and decode in Base64. 6

Coding in base64 6

Decode from base64 7

Python related words 8

Sources 9

Base64 9

Introduction to the program

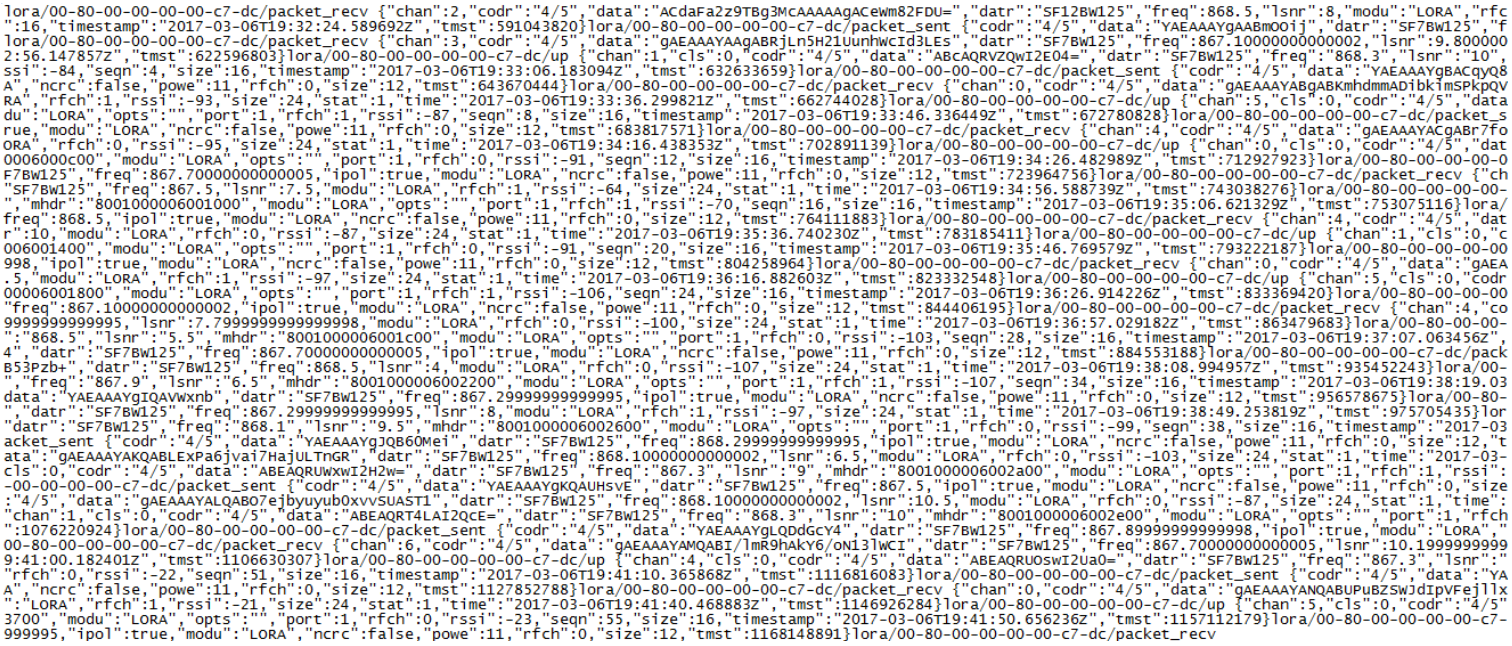
« Decodeb64.py » program is a project initiated by Mr. ROYER, Laurent, from Aubière’s particle physics laboratory, then achieved by Mr. BOURDEAU Etienne, intern from the same lab.

Initially realized in Matlab, it was resumed under Python 3.6.

This program allows to analyze the packets captured by a “Multitech Multiconnect Conduit (MTCDT)” gateway, decode the base64-encoded GPS coordinates, and retrieve other useful information (Signal to Noise ratio, Received Signal Strength Indication, Date / Hour).

Program Analyzing

« Decodeb64.py » program was conceived so it could analyze files like this one:



```
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":"8001000006001000","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":935452243}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:In_file "Data_6mars_cezeaux.txt"

I. Modules import

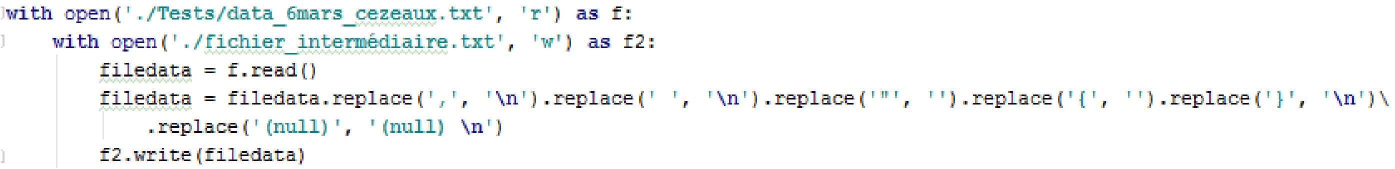
Python 3.6 disposes of a great number of included useful libraries.

```
import base64
import os
```

- ➔ Import of Base64 module, allowing to encode and decode in / from Base64
- ➔ Import of os module, allowing to interact with files

II. Laying out of in_file / Creation of mid_file

In the form of a packet, it is hard to read / understand / analyze information. The program allows to generate an intermediate file, containing the same information, though formatted. This file is automatically destroyed by the program after the chosen information have been extracted.



```
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 – Opening of the in_file, nicknamed « f »

2 – Opening of the mid_file, nicknamed « f2 »

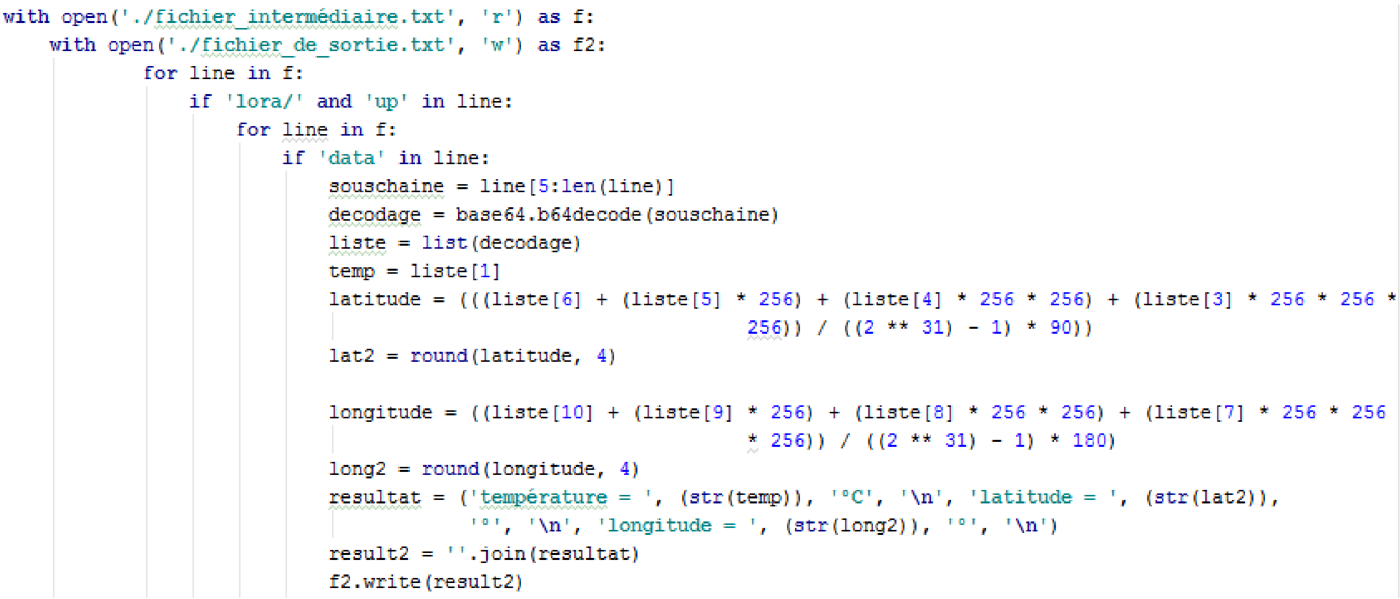
3 – Creation of the “filedata” variable, who reads all the in_file.

4 – From the read file, the variable formats its own content: “,” and “}” become “\n” (“\n” has the value of a line break in code), spaces and “{“are negated (replaced by an empty string), and “(null)” (found in empty packets) are replaced by “(null)\n” so it keeps the “null” information and adds a line break in order to announce the next packet.

5 – Filedata variable’s content is written (formatted content from in_file) in mid_file. This file will be overwritten on each execution.

III. Mid_File Analysis / Base64decode

This part is core to the program, and initially had to be its only functionality.



```
with open('./fichier_intermédiaire.txt', 'r') as f:
    with open('./fichier_de_sortie.txt', 'w') as f2:
            for line in f:
                if 'lora/' and 'up' in line:
                    for line in f:
                        if 'data' in line:
                            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)
                            long2 = round(longitude, 4)
                            resultat = ('température = ', (str(temp)), '°C', '\n', 'latitude = ', (str(lat2)),
                                        '°', '\n', 'longitude = ', (str(long2)), '°', '\n')
                            result2 = ''.join(resultat)
                            f2.write(result2)
```

1 – Opening of the intermediate file (mid_file) previously created, in read mode; nicknamed f.

2 – Opening of the final file (out_file) in write mode, nicknamed f2. This file will be overwritten during every execution.

3 – To ease the reading, I’m going to write the signification of each line of code:

For line in f :	*For each line in mid_file*
If ‘lora/’ and ‘up’ in line :	*If a line is found containing ‘lora/’ and ‘up’*
For line in f:	*For each line from mid_file (from the found pattern)*
If ‘data’ in line :	*if a line is found containing ‘data’*
Souschaine = line[5:len(line)]	*Creation of the ‘souschaine’ sub_string, reading from 6th character* to end of line*
Decodage = base64.b64decode(souschaine)	*Thanks to base64 module, decoding of the sub_string***
Liste = list(decodage)	*Transformation of bytes into list, so the results can be multiplied*
Temp = liste[1]	*Temperature, located in the 2nd part of the list, is isolated in a “temp” variable*

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

Latitude calculation, found on builder’s website, rounded to 4 digits

```
longitude = ((liste[10] + (liste[9] * 256) + (liste[8] * 256 * 256) + (liste[7] * 256 * 256 * 256)) / ((2 ** 31) - 1) * 180)
long2 = round(longitude, 4)
```

Longitude calculation: Same than latitude

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

Laying out of decoded information so they are readable to users.

result2 = ''.join(resultat)	*Join allows to transform a tuple into a string.*
f2.write(result2)	*The result from this loop is written in the out_file.*

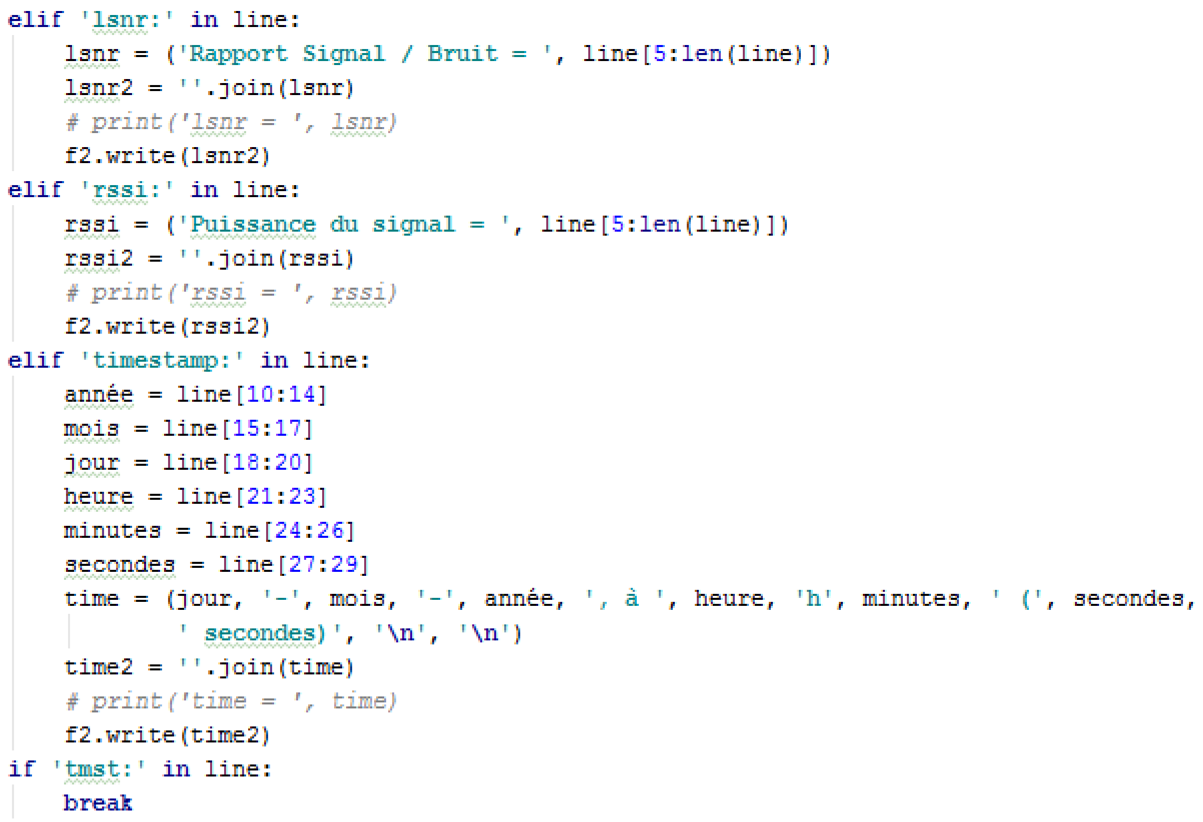
* In Python, counters start at 0, so to select the 6th character, you need to write 5.

** The sub_chain contains only the encoded part in base64. It is decoded in its original form, here in bytes.

IV. Other data recovery

The rest of the data isn’t coded, so it is easier to gather and to write in the out_file. The used processes are the same as the decoding part, from “Transformation of bytes into list […]” until the end.

The program iterates on each read line and checks presence of each pattern thanks to “elif “(else if) operator.

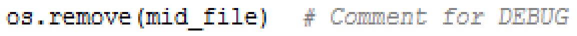


```
elif 'lsnr:' in line:
    lsnr = ('Rapport Signal / Bruit = ', line[5:len(line)])
    lsnr2 = ''.join(lsnr)
    # print('lsnr = ', lsnr)
    f2.write(lsnr2)
elif 'rssi:' in line:
    rssi = ('Puissance du signal = ', line[5:len(line)])
    rssi2 = ''.join(rssi)
    # print('rssi = ', rssi)
    f2.write(rssi2)
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
```

Laying out of temporal data

Loop breaks if ‘tmst’ is met, since each packet finishes by this pattern

V. End of the program



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

Thanks to the « os » module, mid_file is erased. You can comment this line (thanks to the # character) to keep it, so you can check it for debug or to see the full file treated by the loop.

Appendix

Manually code and decode in Base64.

To check if information decoded by the program is right, we can want to decode base64 manually. Here is the method to do so.

Coding in base64

To understand how to decode, we first should understand how to code in base64. Here, our goal will be to code the "a" character.

1 – *Convert the character into ASCII (ASCII table available in appendix). In decimal, in ASCII, a equals **97**.*

a = 97

2 – Convert result into binary:

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

3 – Separate the resulted byte into groups of 6 digits. If the last group isn't complete, add as much 0 as necessary at the right of it until you have a full group.

01100001 = 011000 01(0000)

4 – Convert each group into decimal:

011000 = 24
010000 = 16

5 – Replace these numbers by their equal in base64 alphabet (available in appendix):

24 = Y
16 = Q

6 – A base64 code is always composed from a 4-character group. Here, we only have Y and Q. It is then possible to add up to 2 "=" signs to have a full group. "=" does not have a value.

a = YQ==

Decode from base64

We can now have a better grasp on how to decode from base64.
To do so, we are going to take an easy code: QnJhdm8=

1 – We split up the code in groups of 4:

QnJh dm8=

2 – Let's take the first group: QnJh, then let's convert it in groups of 6 in binary (according to base64 alphabet):

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

3 – Now, let's group back these groups in bytes:

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

4 – Now we can convert each byte into decimal:

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

5 - Converted via the ASCII table, we get:

B r a

6 – Idem for the second group, from the same example:

- dm8= 011101 100110 111100 = (Reminder: "=" has no value)
- in bytes : 01110110 01101111 00000000
- in decimal : 118 111 0
- Decimal to ASCII : **v o**

*7 – Putting results from end-to-end, we get **B r a v o**.*
Decoding is complete.

Python related words

Just like every language, Python owns its own syntax. This part will help you if you don’t have peculiar knowledge about this language.

```
with open('Path/of/file.txt', 'r') as f:
```

Allows to open a text file, and close it automatically after the end of the function / loop, avoiding each part of the program by file.close()

Several opening modes exist, but the most useful are:

- r, for « read ».
- w, for « write ».
- a, for « append ».

The ‘as f:’ part allows to create an alias for the opened file, which helps to call it back later. For example, it will be possible to write ***f.read()***, simpler than ***(‘./Path/of/file.txt’).read().***

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

f.read(), precisely. First, here, let’s make a variable, named filedata, which allows to read the previously opened file. Under Python, 3 modes exist to treat files :

.read(), which reads the entire file
.readline(), which reads the first line of a file
.readlines(), which reads the file line by line and remembers every line in a table.

```
filedata.replace(character_to_replace', 'replacement_character')
```

Here, we call our variable filedata(which still reads our file), and replace a character by another. You can, instead of ‘replacement_character’, let an empty chain ‘’ if you want to delete the character. You can also use ‘\n’ as a replacement character, to put a line break.

```
for line in f:
```

In Python 3.6, the term « line » does not need to be defined, it is already a keyword of the language. It is automatically related to one line of a file. “For line in f” designs automatically “for each line in f”.

```
souschaine = line[start of sub_string(integer only):len(line)]
```

Let’s initialize a variable ‘souschaine’ to extract only one part of the line you want so you can decode it after. We need to tell Python the value of this sub_string, here from the 6th character of the line until it’s end (len(line) means the length of the line).

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

In Python 3.6, only strings can be written in a file. In Decodeb64.py, results are formatted with several strings, forming a ‘tuple’. ‘’.join allows to transform the given tuple into a string, so you can write it in out_file.txt.

Sources

Base64

Understanding base64:
http://forums.devshed.com/python-programming/958996-trying-understand-decode-base64-post2927595.html

How to get latitude and longitude:
http://www.multitech.net/developer/software/dot-box-and-evb-software/survey-gps/

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

ASCII Table: (French)
https://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange

Base64 Alphabet : (French)
https://fr.wikipedia.org/wiki/Base64

What is in MQTT messages:

From builder's site, this link will help you understand what is received in in_file.txt

http://www.multitech.net/developer/software/lora/lora-network-server/mqtt-messages/