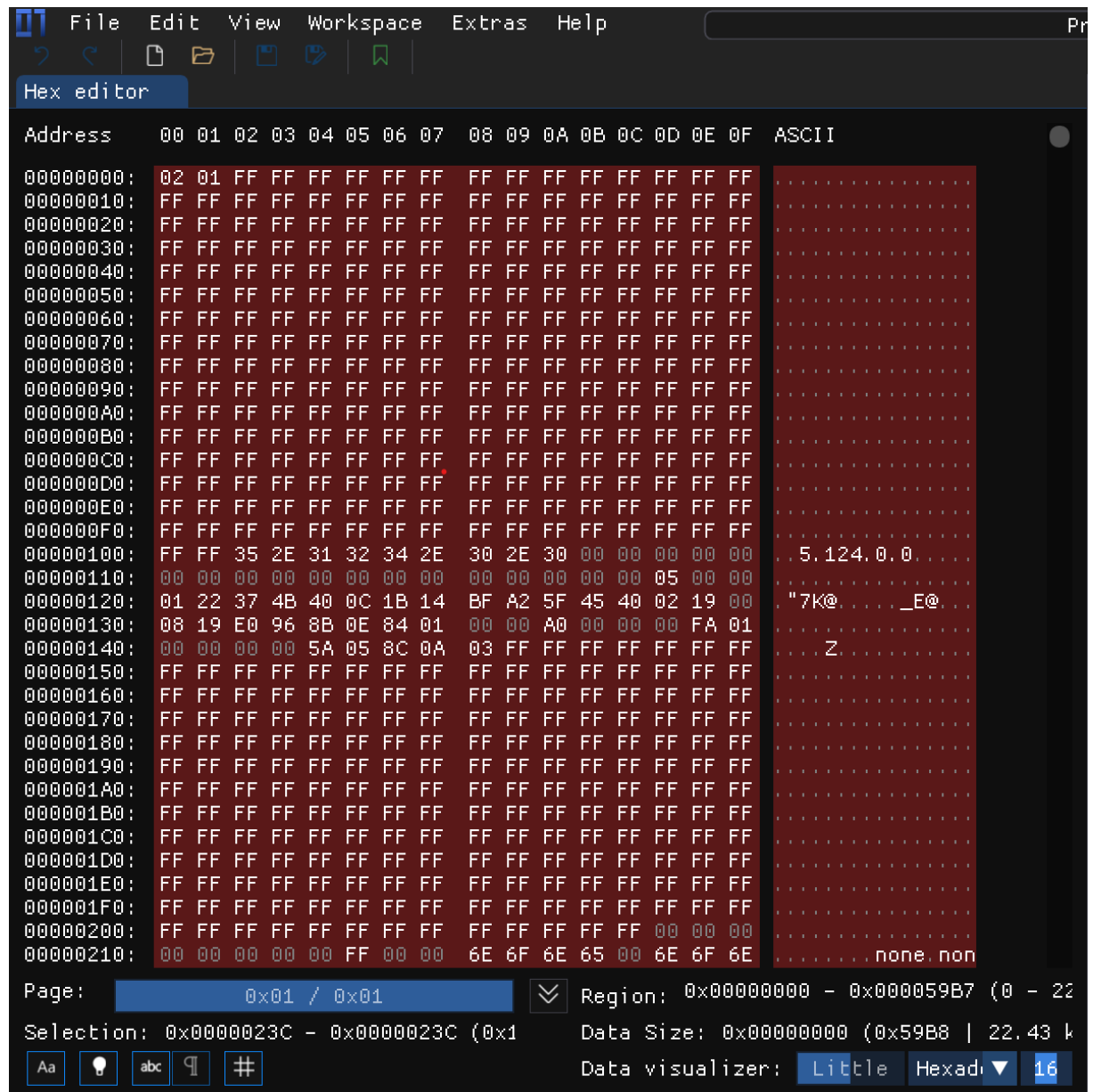


## TP 2 – EL ARIF AMINE

### I- Lire le fichier en ImHex



La première partie était de définir les parties intéressantes dans le fichier binaire et pour faire cela on a défini une struct

```
struct fileStruct{  
  
char metadata[0x248];  
char data[0x56E6] ;  
char endOfFile[0x8A] ;  
  
};
```

La **meta-data** c'est la première partie du fichier contenant des informations sur le fichiers.

Le **data** est le contenu du fichier qu'on cherche à décoder.

Le **endOfFile** est la partie finale du fichier qui commence à partir de STOP.

On suite on passe au contenu de data, on cherche à trouver les coordonnées du GPS (latitude, longitude), grâce à l'indice donné par nos profs, j'ai pu définir cet partie on utilisant le struct suivant :

```
struct dataframe
{
    u8 offset[26];
    u8 longitude[4];
    u8 latitude[4];
    u8 offsetTwo[15];
};
```

Cet partie va se répéter tout au long du partie data, grâce à cet partie :

```
dataframe frame[454] @ 0x248;
```

et finalement on doit définir la structure en disant que ca commence à partir de l'adresse 0x00 :

```
fileStruct var @ 0x00;
```

résultat :

Hex editor																	ASCII
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000250:	01	F0	03	F8	AF	01	F1	03	F8	AF	01	F1	03	F8	9F	01	.....
00000260:	F1	03	8F	FC	52	1B	35	F5	DD	01	00	00	61	01	08	2A	...R.5...a..*
00000270:	65	00	CE	00	12	01	26	1B	00	EF	F8	AF	01	F1	03	F7	e.....&.....
00000280:	AF	01	F0	03	F8	9F	01	F1	03	F8	AF	01	F1	03	F8	9F	.....
00000290:	01	F0	03	98	FC	52	1B	3B	F5	DD	01	00	00	61	01	07	...R.;...a..
000002A0:	2A	65	00	CE	00	2E	01	26	1B	00	EF	F8	AF	01	F1	03	*e.....&.....
000002B0:	F7	8F	01	F0	03	F8	BF	01	F1	03	F7	9F	01	F0	03	F9	.....
000002C0:	AF	01	F1	03	93	FC	52	1B	39	F5	DD	01	00	00	60	01	...R.9.....
000002D0:	06	2A	65	00	DC	00	65	01	26	1B	00	EF	F8	AF	01	F1	.....
000002E0:	03	F7	9F	01	F0	03	F8	AF	01	F1	03	F8	AF	01	F1	03	u8 latitude 57 (0x39)
000002F0:	F8	AF	01	F1	03	85	FC	52	1B	3A	F5	DD	01	00	00	60	...R.t.....
00000300:	01	06	2A	65	00	C6	00	57	02	26	1B	00	EF	F8	9F	01	*e...W.&.....
00000310:	F0	03	F7	AF	01	F2	03	F8	9F	01	F0	03	F8	AF	01	F1	.....
00000320:	03	F7	9F	01	F1	03	79	FC	52	1B	3F	F5	DD	01	00	00	...y.R.?.....
00000330:	60	01	05	2A	65	00	C6	00	3C	03	26	1B	00	EF	F9	9F	*e...<.&.....
00000340:	01	F0	03	F6	AF	01	F1	03	FA	8F	01	F0	03	F7	BF	01	.....
00000350:	F2	03	F9	7F	01	F0	03	73	FC	52	1B	3B	F5	DD	01	00	...s.R.;.....
00000360:	00	60	01	05	2A	65	00	C6	00	0D	04	26	1B	00	EF	F7	*e.....&.....
00000370:	AF	01	F0	03	F7	AF	01	F0	03	F9	9F	01	F0	03	F8	BF	.....
00000380:	01	F1	03	F7	8F	01	F1	03	6F	FC	52	1B	36	F5	DD	01	...o.R.6.....
00000390:	00	00	60	01	05	2A	65	00	C6	00	C5	04	26	1B	00	EF	*e.....&.....
000003A0:	F8	BF	01	F0	03	F9	9F	01	F3	03	F6	BF	01	F1	03	F7	.....
000003B0:	9F	01	F0	03	F8	AF	01	F1	03	5D	FC	52	1B	27	F5	DD	...].R.'.....
000003C0:	01	00	00	5F	01	04	2A	65	00	C6	00	75	05	26	1B	00	..._*e...u.&...
000003D0:	EF	F8	AF	01	F2	03	F7	BF	01	F0	03	F6	BF	01	F1	03	.....
000003E0:	F9	6F	01	F1	03	F8	BF	01	F0	03	49	FC	52	1B	1D	F5	...o.....I.R...
000003F0:	DD	01	00	00	5F	01	04	2A	65	00	C6	00	05	06	26	1B	..._*e...&.....
00000400:	00	EF	F5	9F	01	F0	03	FB	BF	01	F3	03	F5	9F	01	F2	.....
00000410:	03	F9	AF	01	EF	03	F6	AF	01	F2	03	4B	FC	52	1B	1A	...K.R.....
00000420:	F5	DD	01	00	00	5F	01	04	2A	65	00	C6	00	75	06	26	..._*e...u.&...
00000430:	1B	00	EF	F8	AF	01	F0	03	F9	AF	01	F1	03	F6	AF	01	.....
00000440:	F1	03	F8	AF	01	F1	03	F8	8F	01	F1	03	43	FC	52	1B	...C.R.....
00000450:	01	F5	DD	01	00	00	5F	01	04	2C	65	00	C6	00	E5	06	..._*e...&.....
00000460:	1B	11	00	EF	F7	BF	01	F0	03	F8	AF	01	F1	03	F7	9F	.....

## II- Script python pour lire les en csv

```
with open('13caaf25c97e29c8b45ae2e4f9f8ac3b_T0_2e526', 'rb') as file:
    file.seek(0x248)
    trames_bin = file.read()
    trames_size = len(trames_bin)
    trames_count = trames_size // frame_struct.size
    trames=[frame_struct.unpack(trames_bin[i*frame_struct.size:(i+1)*frame_struct.size]) for i in range(trames_count)]
```

Cette partie du code lit le fichier binaire T0, se déplace à la position 0x248 (ignore la méta-data), lit le reste du fichier, calcule la taille et le nombre de trames, puis décode chaque trame du fichier.

```
with open('gpsFile.csv', 'w', newline='') as csvfile:
    fieldnames = ['Latitude', 'Longitude']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    writer.writeheader()
```

ensuite je crée un fichier CSV nommé 'gpsFile.csv', définit les en-têtes de colonnes comme 'Latitude' et 'Longitude', puis j'écris ces en-têtes dans le fichier.

```
for trame in trames:
    latitude = '{:.6f}'.format(trame[1] / 10000000)
    longitude = '{:.6f}'.format(trame[2] / 10000000)

    writer.writerow({'Latitude': latitude, 'Longitude': longitude})
```

je formate ensuite les coordonnées à la bonne format.

Résultat! :

### Leaflet Maps output

Your GPS data has been processed. Your Leaflet Map should be displayed below, and it's also **temporarily** available at [GPSVisualizer.com](https://gpsvisualizer.com). If something doesn't look like you expected it to, please [contact me](#) and explain the problem.

