18/05/2015

Lora-gps capture file format

The acquisition data are stored in csv text files.

Fields separator: comma (‘,’)
Decimal separator: point (‘.’)

Example:

```
43,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,66,70b3d5499ea1e804,0x38,0x38,True,298762
9944,-60,29,7,0xd4
44,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,67,70b3d5499ea1e804,0x5a,0x5a,True,298969
8971,-59,23,7,0x73
45,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,68,70b3d5499ea1e804,0x56,0x56,True,299176
7993,-60,25,7,0x5b
46,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,69,70b3d5499ea1e804,0x34,0x34,True,299383
7016,-59,28,7,0x92
47,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,70,70b3d5499ea1e804,0xb4,0xb4,True,299590
6031,-60,24,7,0x9e
48,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,71,70b3d5499ea1e804,0xd6,0xd6,True,299797
5064,-59,27,7,0x2e
49,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,72,70b3d5499ea1e804,0x70,0x70,True,300004
4079,-59,27,7,0x77
50,70b3d54993208829,2017-05-
18T16:12:17.000Z,46.06799,13.24283,101.01,73,70b3d5499ea1e804,0x12,0x12,True,300211
3216,-59,27,7,0x14
```

Fields

N.	Field	Example
1	n row acquired	53
2	LoraMac of the receiver (logger)	70b3d54993208829
3	gps time	2017-05-18T16:12:17.000Z
4	latitude position	46.06799
5	Longitude position	13.24283
6	altitude	101.01
7	n. of message sent by transmitter with lora protocol	76
8	LoraMac of lora transmitter (8 bytes)	70b3d5499ea1e804
9	crc8 calculated on pyload of message sent by transmitter (fields 7 and 8)	0xe1
10	crc8 calculated by receiver (hex)	0xe1
11	True if fields 9 and 10 are equals. False otherwise	True
12	lora message timestamp	3008319172
13	rssi of lora message received	-59
14	snr of lora message	30
15	sf of lora message	7
16	crc8 of csv row, until field n.15 (hex)	0x5f

Functions used to calc crc8

```
def calc(incoming):
    # convert to bytearray
    hex_data = incoming.decode("hex")
    msg = bytearray(hex_data)
    check = 0
    for i in msg:
        check = AddToCRC(i, check)
    return hex(check)

def AddToCRC(b, crc):
    if (b < 0):
        b += 256
    for i in range(8):
        odd = ((b^crc) & 1) == 1
        crc >>= 1
        b >>= 1
        if (odd):
            crc ^= 0x8C # this means crc ^= 140
    return crc
```