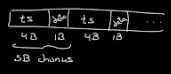
① Activity data → Accel avg over last min
[0-254] → 2B - 1bit free. Used?

② Max size of records → 50400B — Currently: 10.080 datapoints / 7 days

③ Data uploaded on demand

④ Data stored in flash

⑤ Data format




**Assumptions**

A
1. 4B ts → Epoch
2. ts increases strictly every minute.
3. Only included "light" compression methods due to limited resources. Consider Arimatic coding if more compression is desired.
4. Requested data will be a series of contiguous ts.

**Problems**

? 1. What happens when ts updates and a previous value gets overwritten?

**T** How to compress the transmitted data?

① Study data
② Investigate ideas for compression (& how to decompress)
③ Include edge cases/problems & solutions
④ Create file structure & necessary functions

## LC lossless compression techniques

### RLE Run-lenght encoding

Replaces repeated values by value and # repetitions
+ Simple implementation
- Only useful for repeated patterns

Given ts always changes & activity is only 1B, RLE will not provide meaningful improvements.

[250, 250, 250, 248, 248, 249, 249]
↓ RLE
[250, 3, 248, 2, 247, 1, 249, 1]

### ΔE Delta encoding

Replaces values by initial value and the difference between consecutive values.

+ Simple implementation
- Needs predictable, smooth transitions

This encoding would be really useful for the ts, assuming A2. Again, effectiveness for the activity data will depend on the swings in values.

[100, 101, 102, 103, 103]
↓ ΔE
[100, 1, 1, 1, 2]

### HE Huffman encoding

Create a variable-lenght prefix code, where no code is a prefix of another code.

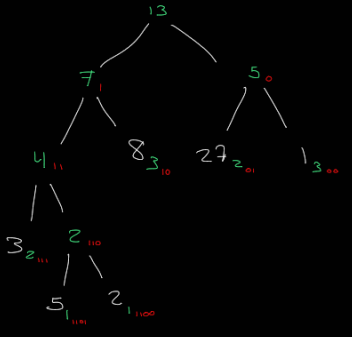
+ Efficient when there are significant variations in symbol frequency
- Needs to include map in compressed data or sent separately

+ map → < 4B
→ 13B

[1, 5, 3, 8, 8, 1, 27, 3, 1, 8, 27, 2]

00 1101 111 10 10 00 01 111 00 10 01 1100

| # | f | bin |
|---|---|---|
| 1 | 3 | 00 |
| 5 | 1 | 1101 |
| 3 | 2 | 111 |
| 8 | 3 | 10 |
| 27 | 2 | 01 |
| 2 | 1 | 1100 |



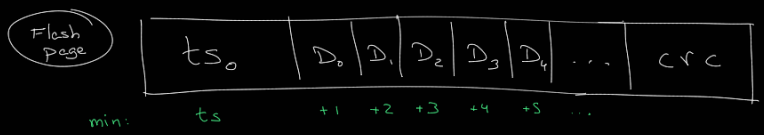

If the previous compression techniques are not feasible, using HE can provide better results with a higher complexity and resources toll.

## FM Frame mods

① Use initial date only. Increment 1 min for every pos increase when reading frame.

+ Allows up to (([page_size] - 4B - crc) × [page_nr])B of data to be stored. → Almost 4x more data density
- Need to account for every time device is rebooted / skips a datapoint.

We can include a datapoint at the beginning of every flash page to account for lost data, rebooting, etc. also including a crc at the end of each page. Although if page is not full, we should still be able to retrieve the data.




Flash page initial ts data should be stored in NV memory for easy access to previous data.