NTAG SmartSensor

NHS31xx data storage

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

Parameters

Data storage is influenced by the setup of and the decisions made in a number of topics:

- EEPROM
- FLASH
- Retrieval
- Data type
- Timestamp
- Storage scheme
- Compression
- SDK support

Storage

EEPROM

- ~3.6 kB available (58 rows)
- Not active by default
- Endurance >10k write cycles
- Cannot be locked
- Writing takes 3 ms / page (64 bytes)
- Write cost of 300 µA / MHz
- Abstraction using EEPROM driver `eeprom_nss`

FLASH

- (30 kB - program size) available
- Active by default
- Endurance >10k program/erase cycles
- Can be locked per sector (1 kB)
- Erase takes 100 ms
  Program takes 1 ms / page (64 bytes)
- Program cost of 500 µA / MHz
- Abstraction using IAP driver `iap_nss`

Compression of temperature values – 1

Temperature sensor

- 0.3ºC absolute temperature accuracy between 0ºC and 40ºC
- 0.5ºC absolute temperature accuracy between -40ºC and 85ºC

Size of one temperature value

- Depends on the required resolution of the measurement when stored.
- ! This is *not* the sensor accuracy!
  - Example: temperature sensor returns 5.24ºC. The real temperature is 5.24ºC +- 0.3ºC. This can be stored using a 0.1ºC resolution as 5.2ºC

Compression of temperature values – 2

Temperature range

Limited by

- IC specification: -40ºC … 85ºC
- Battery characteristics: *e.g.* -20ºC … 65ºC
- Specific use case: *e.g.* -2ºC … 12ºC for a required cold chain environment between 2ºC and 8ºC

The combination of range and resolution results in a limited number of possible temperature values.

Example: With a range between 0ºC and 10ºC and a resolution of 0.1ºC, there are 101 possible values.

A mapping between a temperature value and a number then reduces the required storage space for one temperature value.

Example: for 101 possible values, 7 bits are required per value.

Compression of temperature values – 3

Use case range	Storage resolution	#distinct values	Required bit size per value	Total storage capacity	
				Fits in 24 KB:	Fits in 20 KB:
-40ºC … 85ºC	0.1ºC	1251	11	17455	14545
	0.2ºC	626	10	19200	16000
	0.5ºC	251	8	24000	20000
-20ºC … 65ºC	0.1ºC	851	10	19200	16000
	0.2ºC	526	10	19200	16000
	0.5ºC	171	8	24000	20000
0ºC … 10ºC	0.1ºC	101	7	27429	22857
	0.2ºC	51	6	32000	26667
2ºC … 8ºC	0.1ºC	61	6	32000	26667
	0.2ºC	31	5	38400	32000

Compression of generic data (not limited to temperature)

Other techniques to reduce required storage space

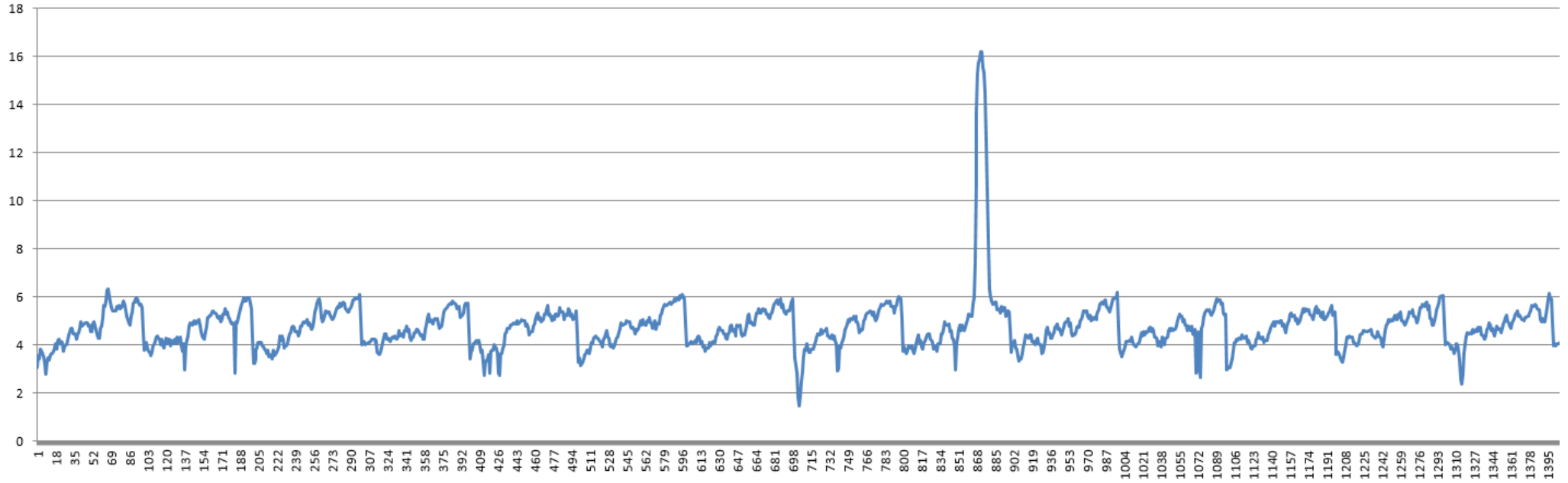
- Reconstruct timestamps using the start time and the sequence number. The sequence number can be recovered based on the physical location in the memory assigned for data storage.

- Use a lossless use case agnostic compression.
  - Huffman coding
  - Run-length encoding
  - Delta encoding
  - Specialized algorithms (e.g. heatshrink)

- Use a lossy compression – use case specific
  - Zone-coding

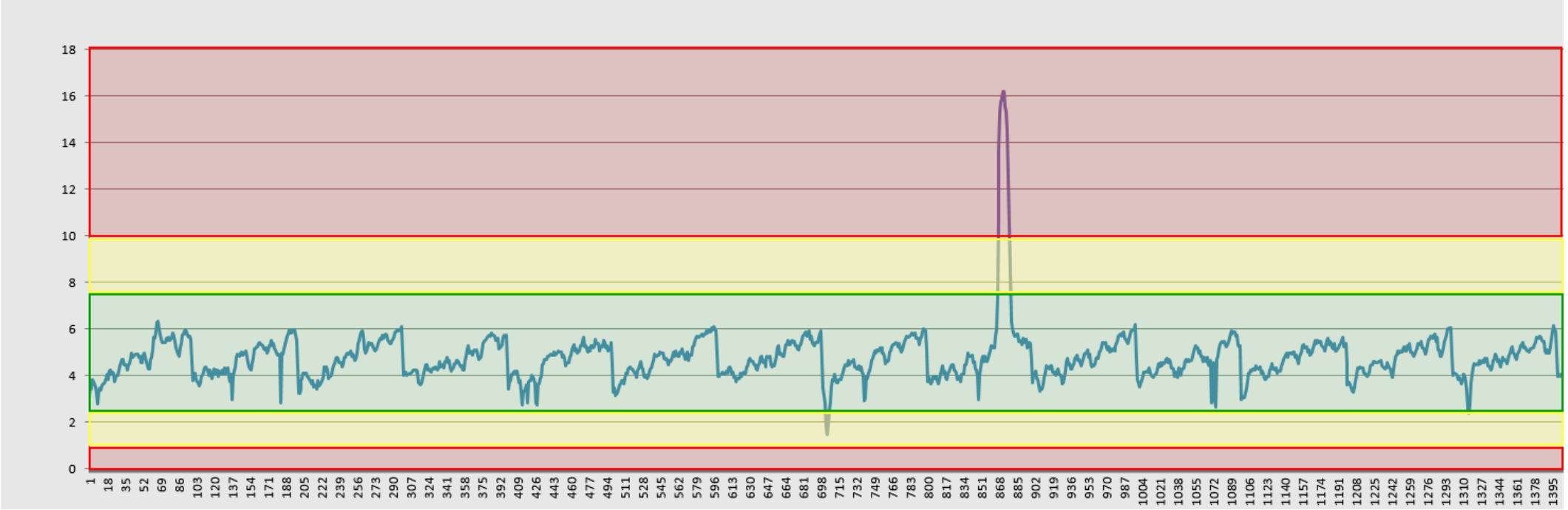
Compression of generic data – zone-coding – 1

- Identify zones, and treat data in each zone differently.
- As an example, consider a product that is kept in a refrigerator, where temperature values have been sampled as per below:



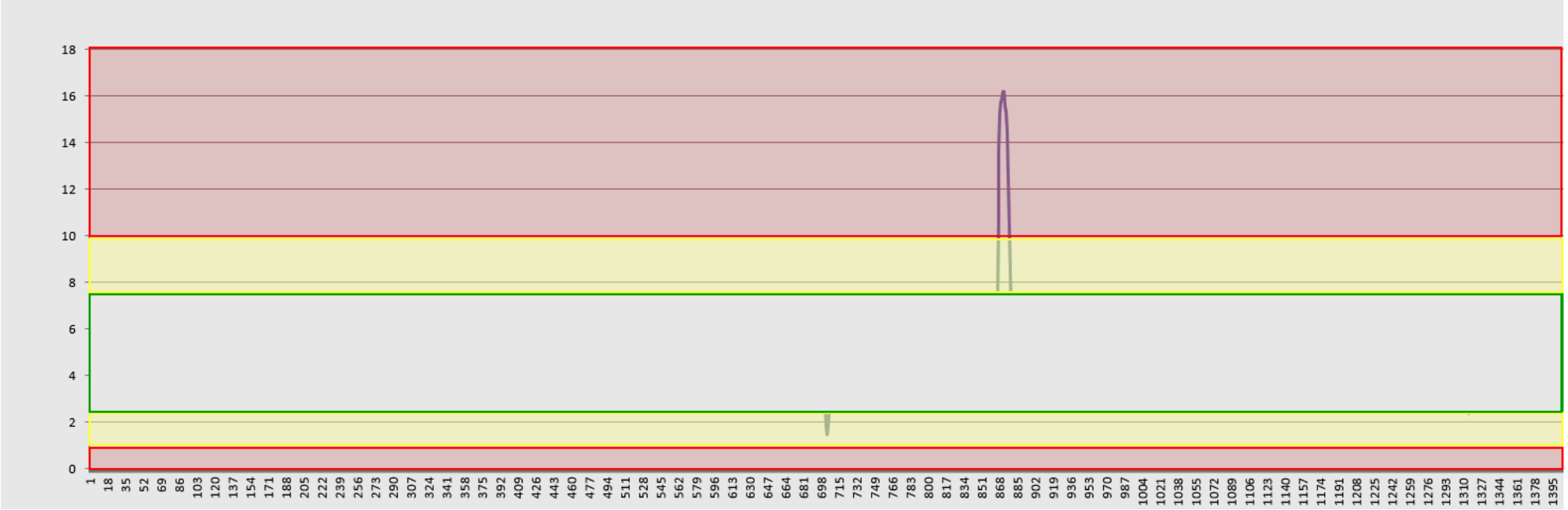
Compression of generic data – zone-coding – 2

- Divide the full temperature range in different bands.
- This is specific per type of product the tag is attached to.
- At least a ‘safe’ zone and a ‘red’ zone – leading to three bands, but more zones can be defined. Below also a ‘danger’ zone is defined – leading to five bands.



Compression of generic data – zone-coding – 3

- As an extreme: only store the outliers.
- Add an index per outlier for reconstruction of the timestamp.
- A middle ground can be to store data in the green zone as well, using a lower resolution.



SDK support

The SDK provides the *storage* module that simplifies storage and retrieval of samples. It is used by the ARM temperature logger firmware demo application.

Benefits:

- Simple API abstracting away all intricacies.
- EEPROM and FLASH characteristics are taken into account
  - Endurance limit will never be reached
  - FLASH Write/Program cost is minimized
- Data is stored packed – no unused bits
- Data can be compressed
  - Algorithm to be provided by the application
  - Can be linked in at compile time
- Direct access to any stored value

SECURE CONNECTIONS
FOR A SMARTER WORLD