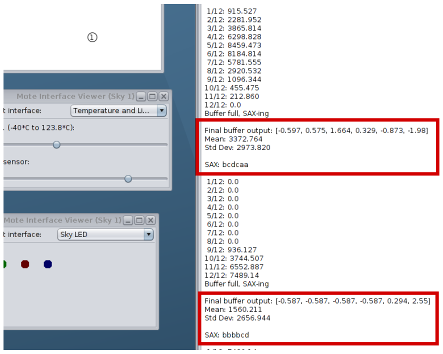

```
 1/12: 915.527
 2/12: 2281.952
 3/12: 3865.814
 4/12: 6298.828
 5/12: 8459.473
 6/12: 8184.814
 7/12: 5781.555
 8/12: 2920.532
 9/12: 1096.344
10/12: 455.475
11/12: 212.860
12/12: 0.0
Buffer full, SAX-ing

Final buffer output: [-0.597, 0.575, 1.664, 0.329, -0.873, -1.98]
Mean: 3372.764
Std Dev: 2973.820

SAX: bcdcaa
 1/12: 0.0
 2/12: 0.0
 3/12: 0.0
 4/12: 0.0
 5/12: 0.0
 6/12: 0.0
 7/12: 0.0
 8/12: 0.0
 9/12: 936.127
10/12: 3744.507
11/12: 6552.887
12/12: 7489.14
Buffer full, SAX-ing

Final buffer output: [-0.587, -0.587, -0.587, -0.587, 0.294, 2.55]
Mean: 1560.211
Std Dev: 2656.944

SAX: bbbbcd
```

Figure 1: Demonstration of SAX aggregation with window size of 2 and alphabet of length 4

1 Description

Symbolic Aggregation Approximation (SAX) was implemented as an in-network data processing technique, compressing the representation while allowing further processing on this symbolic string. Figure 1 shows two rounds of SAX output following data collection, a window size of 2 was used and an alphabet of length 4, i.e the characters `a` through `d` inclusive. 12 C `floats` total 48 bytes of data, this can be reduced by a factor of 4 using `char` representation instead, a window size of 2 halves the number of output samples and lowers the required memory to just 6 bytes.

2 Specification

SAX is implemented in two separate steps, that of transforming the time-series into Piecewise Aggregate Approximation (PAA) representation and then representing this numeric series with a symbolic alphabet.

2.1 PAA

The standard deviation and mean of the data series were first calculated, these are required for Z-normalisation. This normalisation process takes a series of data and transforms it into one with a mean of 0 and a standard deviation of 1. This changes the context of the value from being measured in lux to being a measure of a samples distance from the mean, 0, in standard deviations. This allows comparison of different time-series.

Following Z-normalisation, the size of the series is reduced by applying a windowing function. This takes subsequent equally-sized groups of samples and reduces the group to the mean of those values.

As a result of these two actions, the original time series has been reduced to a given length of samples with a mean of 0 and standard deviation of 1.

2.2 SAX

With the result of the above, the remaining step is to replace each sample value with a symbol to represent it. The amount of symbols to be used is given, each will represent the same probability range when considering a Gaussian distribution of mean 0 and standard deviation of 1. This can be achieved by using standard deviation breakpoints defined such that the area under Gaussian curve between breakpoints is the same.

3 Implementation

The SAX functionality was added as an alternative buffer rotating mechanism over the original 12-to-1/4-to-1/12-to-12 aggregation system. The length of the output buffer is calculated such that it can be allocated. From here the input buffer is Z-normalised using the `normaliseBuffer(buffer)` function from the `buffer.h` header. This function iterates over each value in the buffer, subtracts the buffer's mean and then divides by the standard deviation. Following this, the buffer is aggregated using the same 4-to-1 aggregation function `aggregateBuffer(bufferIn, bufferOut, groupSize)` as the group size is variable. The output from this function represents the PAA form of the initial data series.

This final buffer is handled using `handleFinalBuffer(buffer)` where a pre-processor directive checks whether SAX is being used. If so the PAA buffer is *stringified* using `stringifyBuffer(buffer)` which performs the SAX symbolic representation.