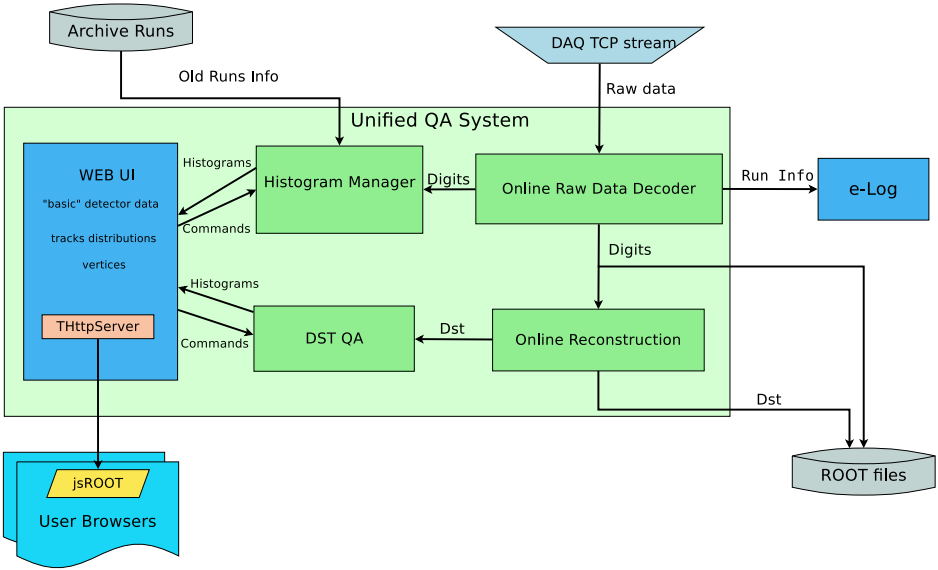
# Status of the Raw Data Decoder and Online Histogramming, Problems and Plans

Ilnur Gabdrakhmanov

Joint Institute for Nuclear Research, Laboratory of High Energy Physics

BMN Detector Council Meeting
June 13, 2023


# General system scheme

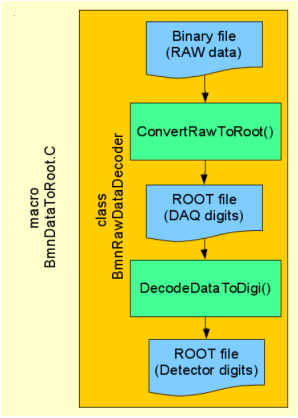


## My zone of responsibility - by gray color

| Detector | Decoding | Reco/HitMaker | Monitoring: Digi | DST |
|---|---|---|---|---|
| Triggers | ✓ | - | ✓ | - |
| Silicon | ✓ | ✓ | ✓ | ✓ |
| SiBT | ✓ | ✓ | ✓ | ✓ |
| GEM | ✓ | ✓ | ✓ | ✓ |
| CSC | ✓ | ✓ | ✓ | ✓ |
| DCH | ✓ | ✓ | ✓ | ✗ |
| ToF400 | ✓ | ✓ | ✓ | ✓ |
| ToF700 | ✓ | ✓ | ✓ (needs extension) | ✓ |
| ScWall | ✓ | ✓ | ✓ | ✓ |
| FHCal | ✓ | ✓ | ✓ | ✓ |
| Hodoscope | ✓ | ✓ | ✓ | ✓ |
| NDet | ✓ | ✓ | ✓ | ✓ |
| Profilometer | ✓ ✗(needs reworking) | - | ✓ (needs refactoring) | - |
| GlobalTracking | - | ✓ | - | ✓ |

- Thanks to the calorimeter group which developed their part of decoding & monitoring.
- Profilometer part implemented in a separate pipeline separated from the main DAQ.

# Decoding scheme (currently under refactoring)




First step (Data Converter):

- Read a binary data file with RAW-data.
- Parse the data blocks: run/spill/event/module.
- Create «DAQ-digits» (ADC, TDC, TQDC, HRB, TTVXS, etc.) accordingly DAQ-data-format and write them into a tree.

Second step (Data Decoder):

- Read detector mappings (channel-to-strip) from the Unified Database
- Calculate pedestals and common modes of channels
- Clear noisy channels
- Decode DAQ-digits into detector-digits (BmnGemDigit, BmnTofDigit, etc.)
- Write the tree with detector-digits to a ROOT-file

# Decoder updates

- Raw data converter:
  - TTVXS, TDC72VXS
  - JSON data blocks
  - Stat event MSC16 processing
  - Raw data converter source as FairSource class inheritor (in order to rewrite the decoding procedure as a task)
- Decoder:
  - SiBT decoder implemented
  - CSC decoder extended to use various local mapping for different stations
  - Profilometer source & decoder implemented (source needs revision)
  - Various optimisations and fixes for strip detectors ADC processing
  - General decoder partly rewritten as a task
  - Strip ADC calibration/noise data saving/loading implemented but exact logic for dealing with consecutive run (or files in a long run) is not clear:
    - How to use/update previous run pedestals/noise channels in an online mode?
    - In an offline production: should we process files of a large run in a queue and store temporary calibration data between them? (seems faster in cpu, but may not fit free space on a cluster machine)
    - Or should we process them separately and use each file's own calibration? (seems more correct, but requires necessary temporary raw root file. How to deal with small files without pedestals?)



BM@N

# Basic QA frontend with hardcoded histograms

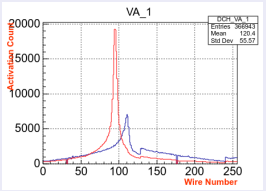
## Implementation details:

- The data processed and transferred from the previous stage is used to fill ROOT histograms. Which in turn are sent to the end users via http.
- CERN jsROOT library is used to transform the ROOT object to the html histograms.
- Base class for histogram sets BmnHist is used in:
  - BmnHistTrigger
  - BmnHistGem
  - BmnHistToF
  - ... ...

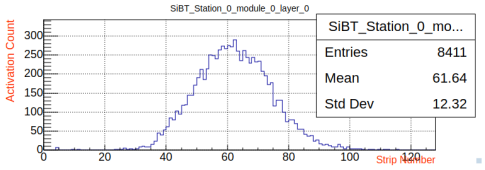
Thus addition of the new detector histogram set is rather simple.

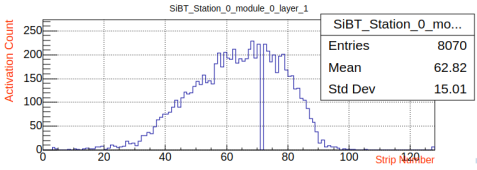
## Reference run:

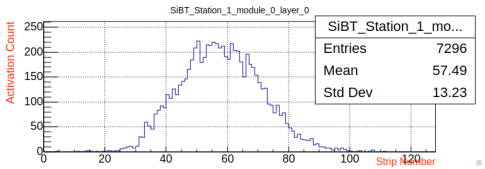
- ✓ Ref run imposition
- ✓ Autoselection of similar runs

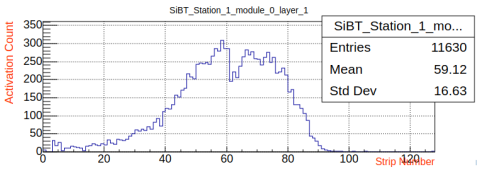


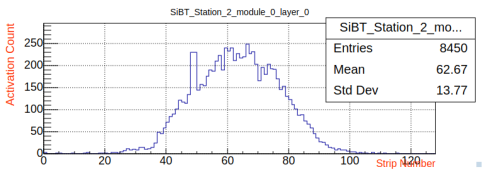
# Live example of SiBT digits online decoding

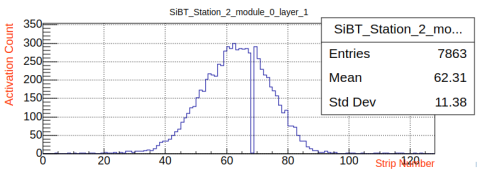


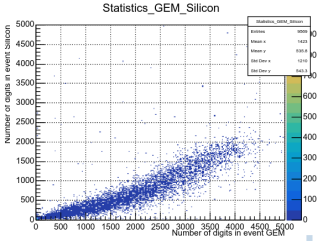



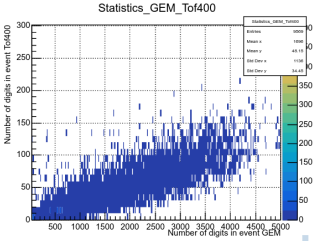



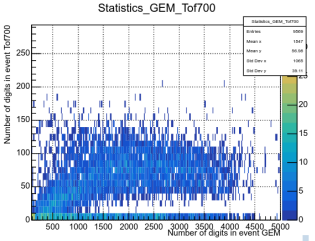



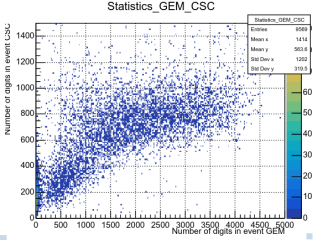

There were some problems with local channel maps at the beginning of the Xe run.

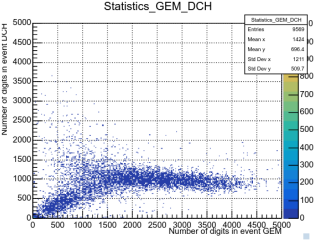
# Live example of detectors correlation

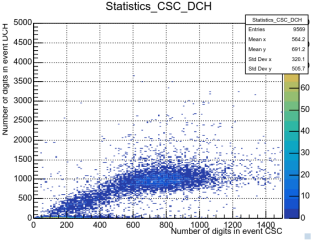


## Existing alternative online processing frameworks

- TDAQ (ATLAS)
  - tightly integrated with other ATLAS software
  - thus it is rather difficult to deploy in other program environment
- FairMQ (GSI FAIR) work by I.Romanov, K.Gertsenberger
  - seems to be quite flexible in deployment and settings (with DDS as an option)
  - but requires additional wrapper code
  - seems not to work in an interactive ROOT macros

## FairRoot way of analysis via FairTask's
(Extensively being used in the BmnRoot)

- FairRunAna - task manager class
- FairSource - abstract class for a data source
- FairSink - abstract class for a data destination manager

## Typical analysis macro workflow:

- BmnFileSource/FairFileSource (input data file )
- Task1 (executed event-by-event)
- Task2
- Task3
- ...
- FairRootFileSink (output data file)

## Simplest way to move existing reconstruction code to online

Less code $\rightarrowtail$ Less errors

ZMQ transfer classes for FairRunAna

- BmnMQSource - ZeroMQ SUB socket[1] based source class
- BmnMQSink - ZeroMQ PUB socket based sink class

### Benefits

- No need to rewrite existing bmnroot analysis code. (No need to touch any working task)
- It became possible to combine several analysis macros by source/sink network interfaces

[1] https://zeromq.org

# BmnRoot QA structure

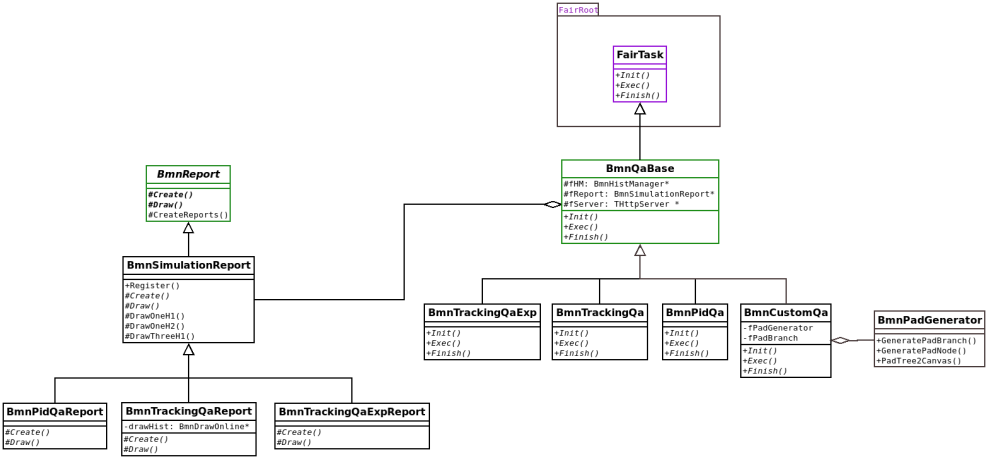
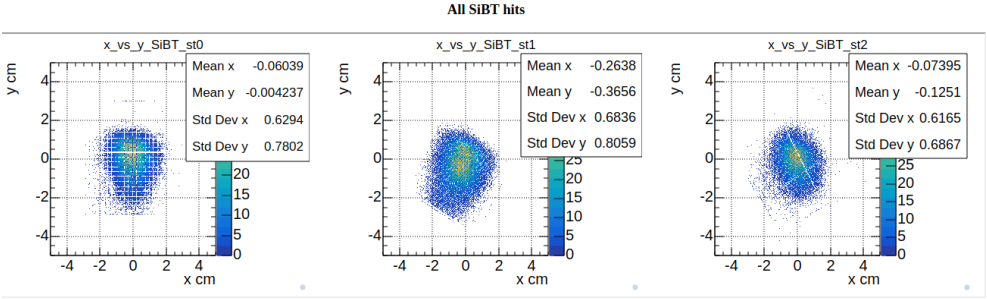
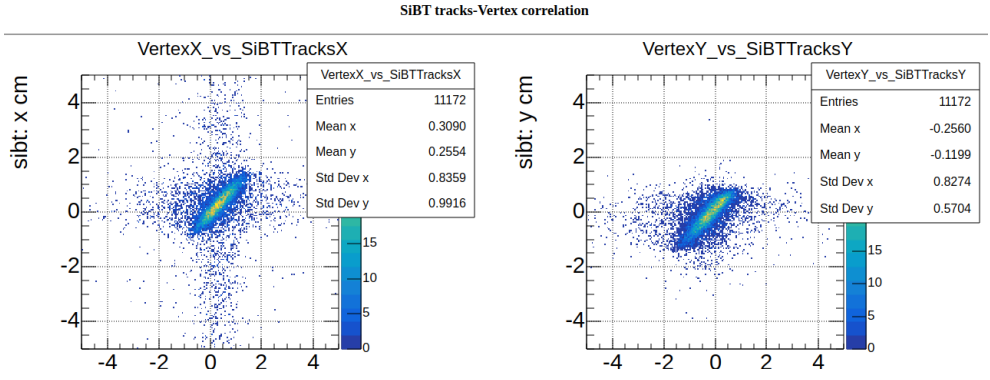



Figure: QA main classes (green ones were forked from CbmRoot)

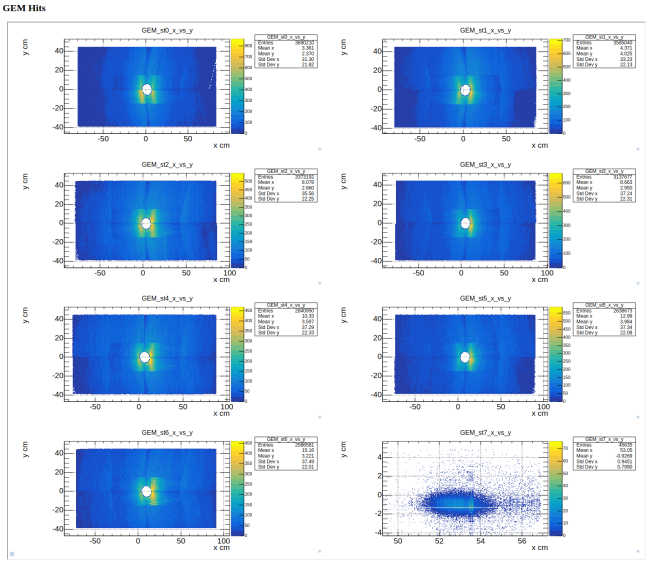
# Live example of SiBT hits online reconstruction



# Live example of SiBT hits vs Vertex coordinates

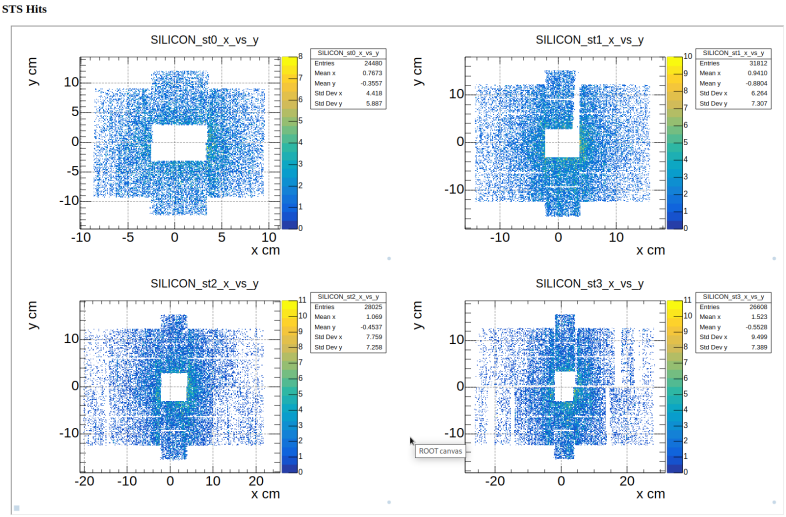


# Live example of GEM hits online reconstruction

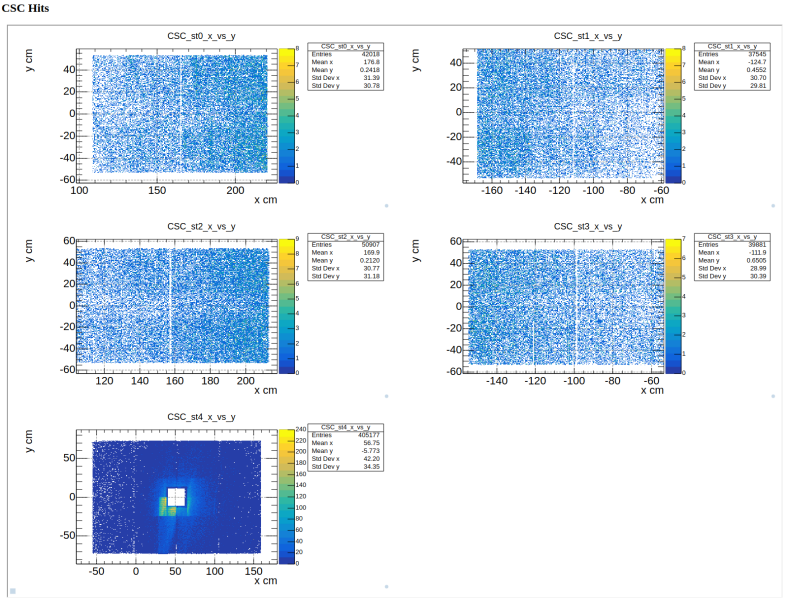



The noise channel detection logic should be reevaluated for strip detectors on the beam

# Live example of FSD hits online reconstruction



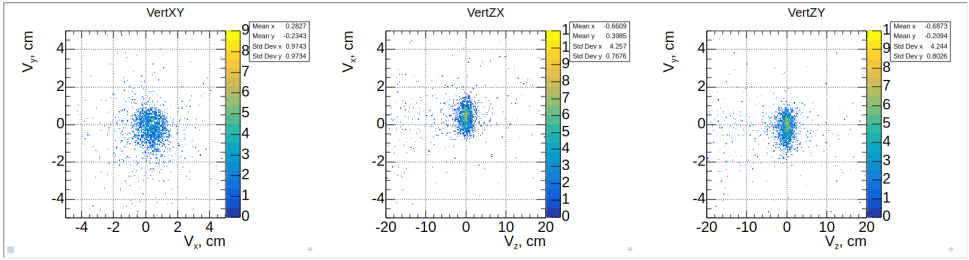
# Live example of the CSC hits online reconstruction



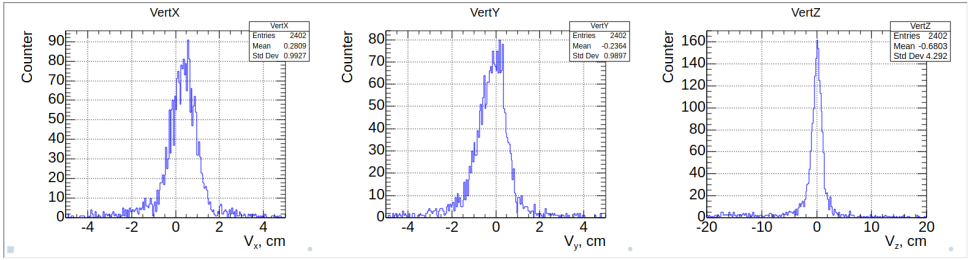
# Live example of the primary vertex online reconstruction

**Vertex profile**

**Vertex profile 2D**




**Vertex profile 1D**



# Custom «no code» histograms. Motivation

## Why?

Experiment upgrade as well as conduction of two experimental setups require distribution of work on the development of the online QA system.
Namely each detector team should be able to extend system's functionality easily.

# Custom «no code» histograms. Motivation

## Why?

Experiment upgrade as well as conduction of two experimental setups require distribution of work on the development of the online QA system.
Namely each detector team should be able to extend system's functionality easily.

## Main objectives:

- Move monitoring configuration outside of the code
- Make addition of histogram simple and flexible (It should not require code rebuild)
- Implement filling logic configurable as well (thanks to ROOT TTree::Draw text parser it was possible)

# Custom «no code» histograms. Motivation

## Why?

Experiment upgrade as well as conduction of two experimental setups require distribution of work on the development of the online QA system.
Namely each detector team should be able to extend system's functionality easily.

## Main objectives:

- Move monitoring configuration outside of the code
- Make addition of histogram simple and flexible (It should not require code rebuild)
- Implement filling logic configurable as well (thanks to ROOT TTree::Draw text parser it was possible)

## Implementation

BmnPadGenerator class - creates a pad structure in the canvas on the basis of json scheme.

Test code example:

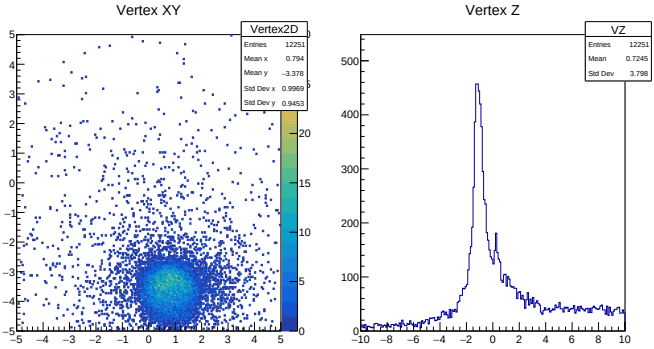
```
BmnPadGenerator *g = new BmnPadGenerator();
g->LoadPTFrom(FileName);
BmnPadBranch * br = g->GetPadBranch();
TCanvas* can = new TCanvas("canHits", "", 1920, 1080);
g->PadTree2Canvas(br, can);
BmnHist::DrawPadTree(br);
```

# Simple configuration

**JSON scheme:**

```
{
    "Name": "Custom canvas",
    "Title": "Custom Canvas",
    "DivX": "2",
    "DivY": "1",
    "Pads": [
        {
            "Class": "TH2F",
            "Name": "Vertex2D",
            "Title": "Vertex XY",
            "Variable": "BmnVertex.fY:BmnVertex.fX",
            "Selection": "(BmnVertex.fZ>-10 && BmnVertex.fZ<10)",
            "Options": "colz",
            "Dimensions": [
                200,
                -5,
                5,
                200,
                -5,
                5
            ]
        },
        {
            "Class": "TH1F",
            "Name": "VZ",
            "Title": "Vertex Z",
            "Variable": "BmnVertex.fZ",
            "Selection": "(BmnVertex.fZ>-10 && BmnVertex.fZ<10)",
            "Dimensions": [
                200,
                -10,
                10
            ]
        }
    ]
}
```

**Canvas structure:**




- Works well for data in TClonesArray branches
- Doesn't work for single object branches out of the box (only with additional code for each class)
- User interface for scheme updating is not yet ready

# QA experience overview. Possible future improvements.

- jsROOT:
  - Sometimes not updating scales in the jsROOT histograms. Even after restart of a QA.
    - A more efficient and stable tool is needed, for example Grafana of VisionForge, but it will require rewriting basic histograms functions
- Decoding:
  - Adequate logic to deal with calibration and noise channel masking is needed for long runs
    - Save/Load procedures are working but
    - Noise channels may change during long series of runs
  - Test data for detectors (cosmic, laser, $\alpha$ source) before the start would be really helpful for channel map problems detection
- DST QA:
  - Slow tracking procedure (even L1)
    - Actually not a problem (the character of distribution stays the same. ZMQ just drops events not fitting the buffer)
    - sometimes gives noticable lag of processing cached events of a previous run for a couple of minutes
  - Needs to fully implement reference runs in the unified QA as well:
    - Maybe we should use UniDB to store reference run sets
  - Needs to finish Custom QA (no code). Particularly implement user interface for schemes.
- SPbU students helped with the work:
  - K. Mashitsin - GEM decoding algorithm improvements and fixes
  - A. Driuk - Digi correlation histograms, SiBT histograms
  - A. Iufriakova - ADC decoder SIMD optimisation

# Conclusion

- Purely online QA system based on BmnHist class
  - currently includes only digits (1st variant of monitoring)
  - not flexible enough
- Offline QA system based on BmnQaBase is being unified for online/offline usage:
  - hardcoded variant currently includes only DST data (2nd variant)
  - Experimental "no code" approach were developed in order to simplify extension of the system. But still not all elements implemented.(3rd variant)
  - ZeroMQ transfer source/sink classes were developed for FairRunManager based analysis.
- Decoder future work:
  - Rewriting as a source/tasks (partly done)
  - Develop logic for calibration/noise long time correct processing
- QA system future work:
  - Frontend possible move to Grafana or VisionForge
  - Automate starting/restarting of a QA components without losing information
  - Possible integration with other online monitoring BM@N projects.

# Conclusion

- Purely online QA system based on BmnHist class
  - currently includes only digits (1st variant of monitoring)
  - not flexible enough
- Offline QA system based on BmnQaBase is being unified for online/offline usage:
  - hardcoded variant currently includes only DST data (2nd variant)
  - Experimental "no code" approach were developed in order to simplify extension of the system. But still not all elements implemented.(3rd variant)
  - ZeroMQ transfer source/sink classes were developed for FairRunManager based analysis.
- Decoder future work:
  - Rewriting as a source/tasks (partly done)
  - Develop logic for calibration/noise long time correct processing
- QA system future work:
  - Frontend possible move to Grafana or VisionForge
  - Automate starting/restarting of a QA components without losing information
  - Possible integration with other online monitoring BM@N projects.

Please send all channel maps before the experiment starts!

# Conclusion

- Purely online QA system based on BmnHist class
  - currently includes only digits (1st variant of monitoring)
  - not flexible enough
- Offline QA system based on BmnQaBase is being unified for online/offline usage:
  - hardcoded variant currently includes only DST data (2nd variant)
  - Experimental "no code" approach were developed in order to simplify extension of the system. But still not all elements implemented.(3rd variant)
  - ZeroMQ transfer source/sink classes were developed for FairRunManager based analysis.
- Decoder future work:
  - Rewriting as a source/tasks (partly done)
  - Develop logic for calibration/noise long time correct processing
- QA system future work:
  - Frontend possible move to Grafana or VisionForge
  - Automate starting/restarting of a QA components without losing information
  - Possible integration with other online monitoring BM@N projects.

Please send all channel maps before the experiment starts!

Thanks for your attention!