



SAMBA: General Purpose DAQ System

Dark Matter UK, 5th May, 2022

Michel Gros, Université Paris-Saclay and Mark Slater, Birmingham University

SAMBA is a general purpose, scalable, user-configurable Data Acquisition System that has been used for Edelweiss, NEWS-G and others

It was originally written and designed by Michel Gros in 2005 with an emphasis on flexibility

It includes the following general features:

- Multiple, user-defined detectors with multiple channels for each

- Multiple ADC/DAC of various types

- Completely configurable setup of an experiment

- Slow Control integration

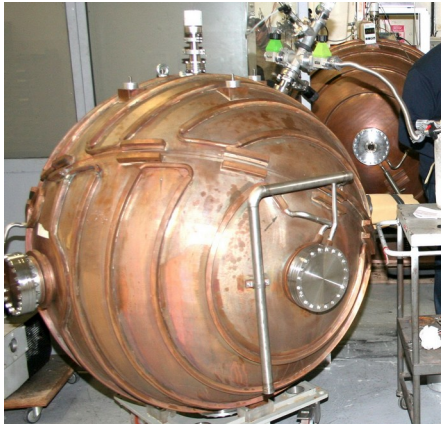
- Continuous data recording or triggering and event reconstruction

- Live, user configurable plotting including histograms, FFT and trigger display

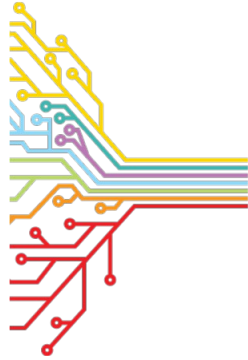
- Scalable to multiple DAQ machines

Typical Experimental Setup

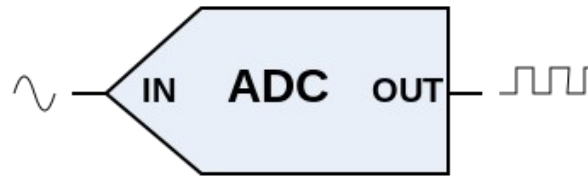
Detector



Readout Electronics



Analog to Digital Converters



Signal Processing





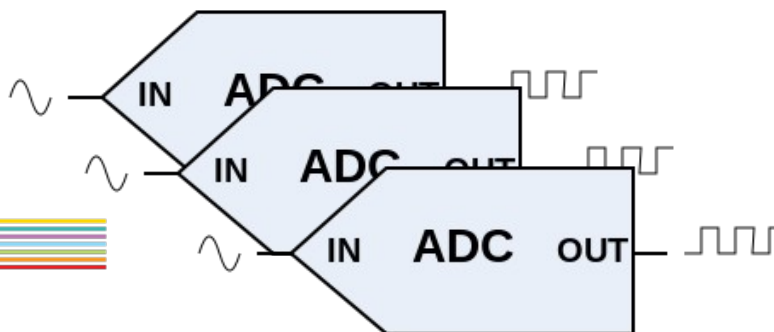
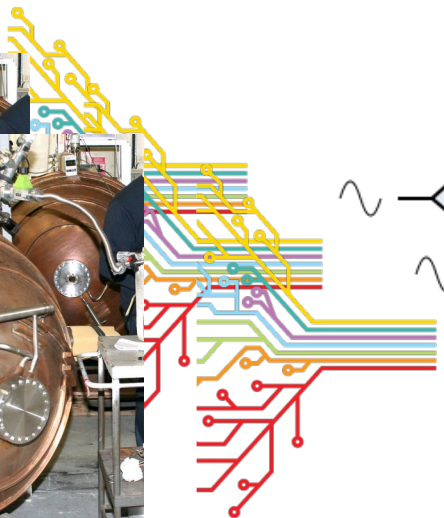
Typical Experimental Setup

Detectors

Readout
Electronics

Analog to Digital
Converters

Signal Processing



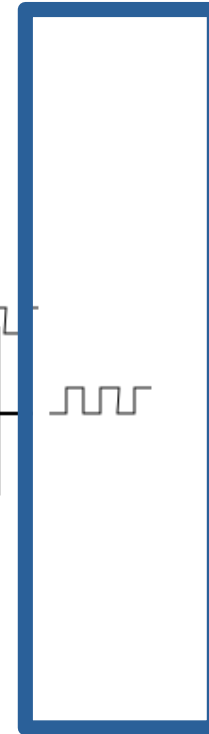
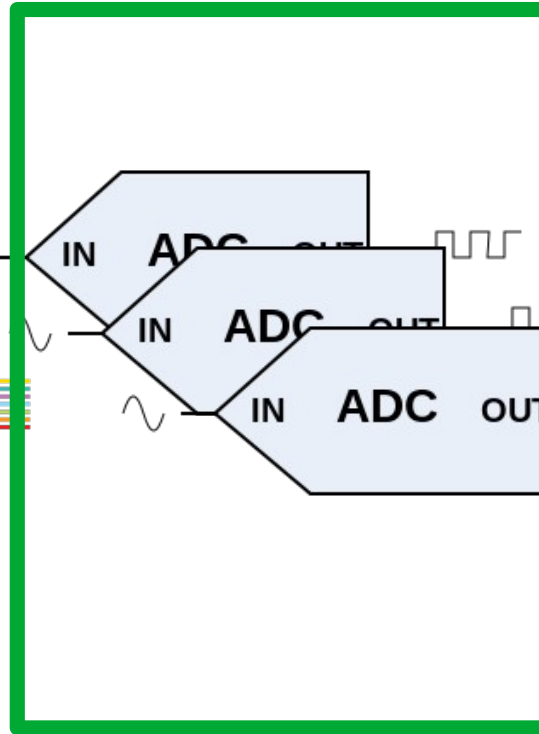
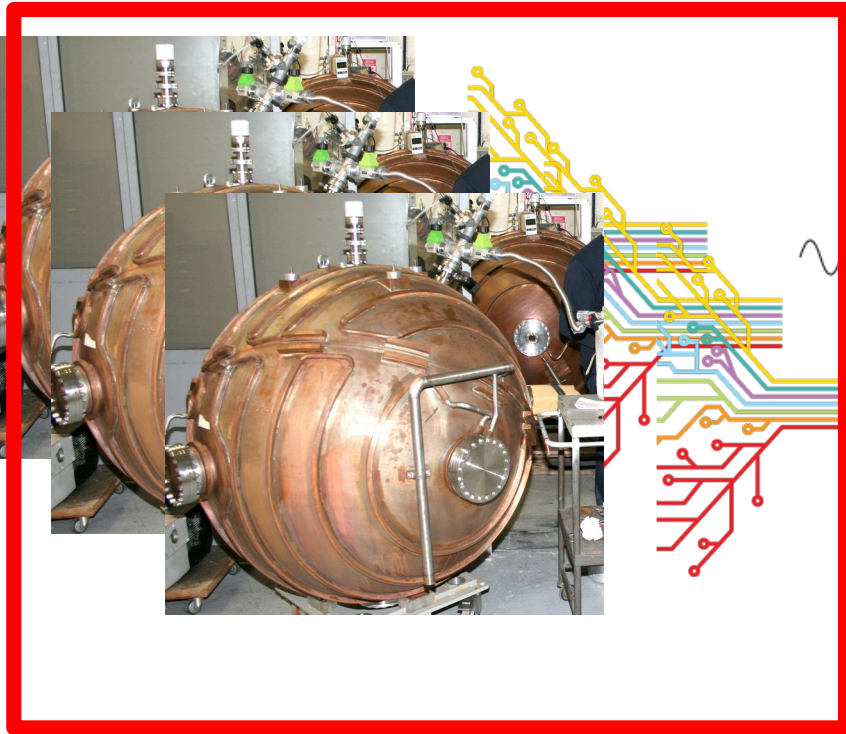
A SAMBA Model

Detectors

Readout Electronics

Analog to Digital Converters

Signal Processing



Detectors

Digitisers

Dispatchers

Acquisition

A SAMBA Configuration (or Model) for a particular experiment describes the following:

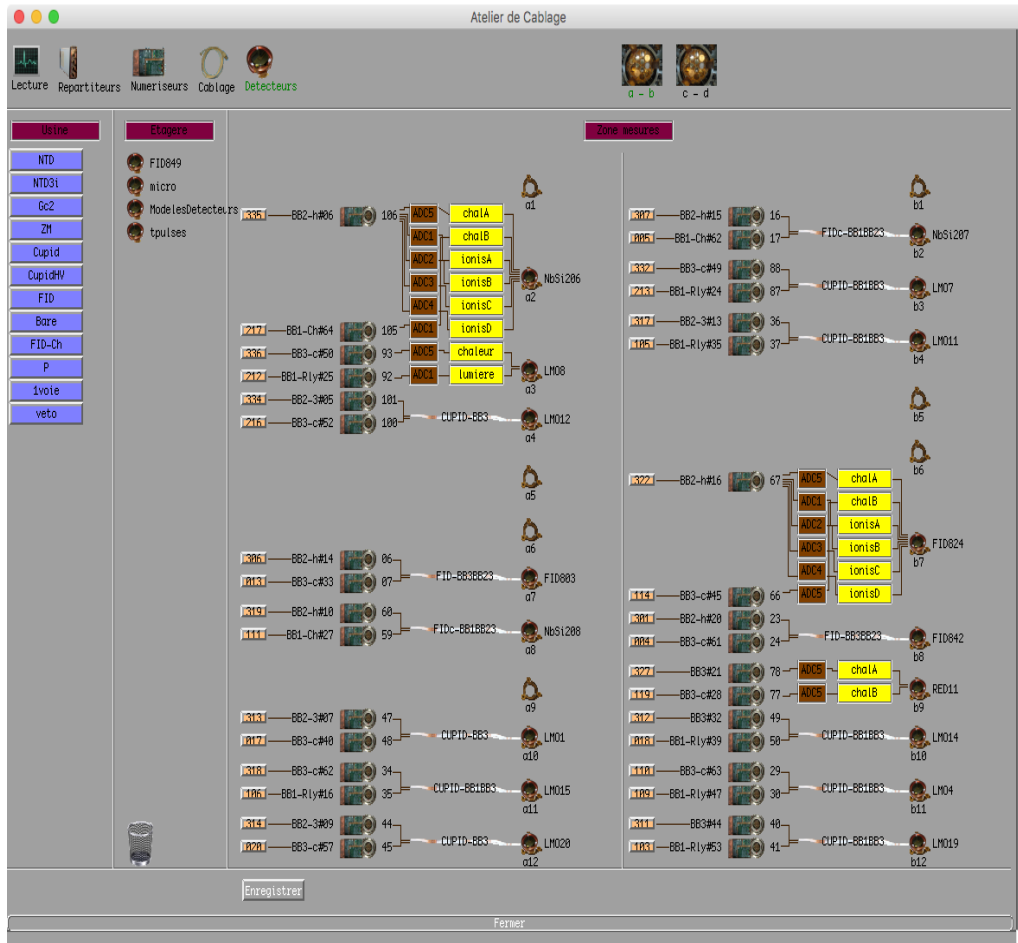
- Detectors
- Digitisers of output signals with appropriate parameters
- Interfaces (dispatchers) that tell SAMBA how to do I/O with the hardware
- Slow Control commands linked to this
- Channels for each of the above
- Signal processing chains (triggering, smoothing, filtering, etc.)
- Plots for Monitoring
- Data Storage

Thanks to the modular design, any unsupported hardware elements such as Dispatchers, etc. can be easily added in

Configurations are stored in human-readable JSON-style formats or can be viewed and edited through the GUI



EDELWEISS Setup



Data coming from channels can be split/combined as required and can be recorded either as a stream or as a series of events and processed using a number of ready-to-use algorithms included with SAMBA

Stream-based algorithms include:

- Filtering (butterworth, chebyshev, elliptic) ●
- Smoothing ●
- Demodulation ●
- Mean value ●

Methods for triggering include

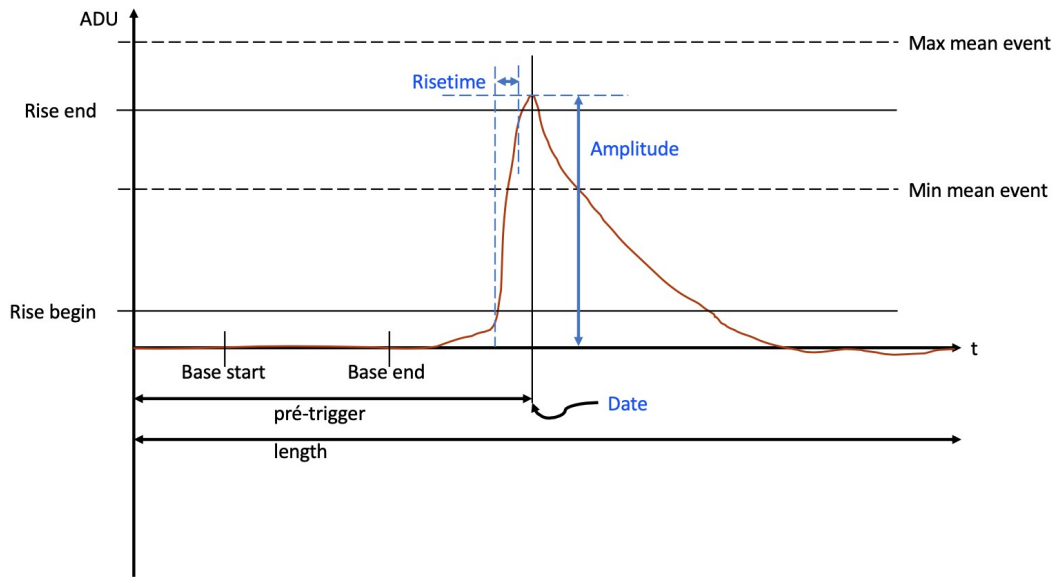
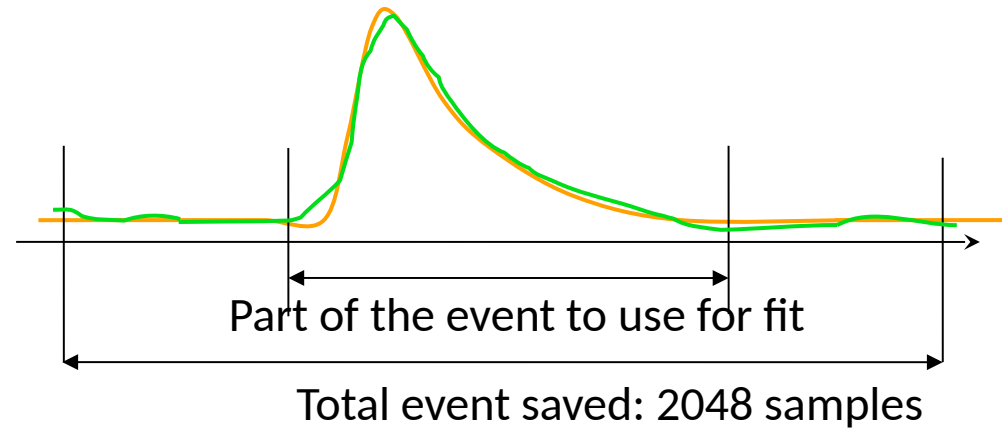
- Threshold ●
- Amplitude+rise time ●
- Derivative ●
- Random ●

Processing, Events and Storage are all user-defined by channel

Event Definition

An adaptive trigger is used for event based recording

A fit to a pulse shape (either analytic or tabular) is then used to define the properties of the event



After the fit has been done, further triggering and filtering can be applied on the fitted values



Event Definition via GUI

Definition evenements

Detecteur **LM01**
 [Reset] [Appliquer]
 [Enregistrer]

	chaleur	lumiere
Definition	particuliere	particuliere
Duree (millisecon.)	2048	2048
Pre-trigger (%)	50	50
Temps mort (millisecon.)	1000	1000
Caracterisation		
Debut base (% pretrig)	25	25
Fin base (% pretrig)	75	75
Debut montee (% ampl.)	90	90
Fin montee (% ampl.)	0	0
Fenetre recherche (millisecon.)	10	10
Decalage (millisecon.)	0	0
Template		
Dimension (points)	128	128
Montee (millisecon.)	15	5
Descente1 (millisecon.)	30	5
Descente2 (millisecon.)	500	12.6
Facteur2	0.2	0.699
Integrales		
Debut rapide (millisecon.)	0	0
Duree rapide (millisecon.)	0	0
Debut longue (millisecon.)	0	0
Duree longue (millisecon.)	0	0
Calibration		
Min evt moyen (ADU)	8000	8000
Max evt moyen (ADU)	0	0

[Reset] [Appliquer]
 [Fermer]

Trigger

Detecteur **LM01** [Enregistrer]
 [Reset] [Appliquer]

chaleur

chaleur **particuliere**
 [Reset] [Appliquer]

Detection
 Trigger **random**
 Sens du pulse **indifferent**
 [Reset] [Appliquer]

Taux (Hz) **0.0063**
 [Reset] [Appliquer]

Coupages
 Plancher ampl. **0**
 Plafond ampl. **0**
 Montee mini (ms) **0**
 Montee maxi (ms) **0**
 Bruit max **99999.9**
 Duree mini (ms) **0.0063**
 [Reset] [Appliquer]

-calcul -neant evt moyen

lumiere

lumiere **particuliere**
 [Reset] [Appliquer]

Detection
 Trigger **neant**
 Sens du pulse **negatif**
 [Reset] [Appliquer]

Coupages
 Plancher ampl. **0**
 Plafond ampl. **0**
 Montee mini (ms) **0**
 Montee maxi (ms) **0**
 Bruit max **99999.9**
 Duree mini (ms) **0.027**
 [Reset] [Appliquer]

-calcul -neant evt moyen

[Fermer]



Typical Data Taking

07:41

Mode **events** Running Stop

Regulation **off**

Archiving **off**

Save conditions

Normal data taking in progress

Sessions **1**

Id	run	duration	evts seen	saved
loc	neant	50,483 s	509	0

Histos reset k30 **en route**

Compensation

Save detectors

Stop dispatcher

Events



Event **509** detector **sphere**

t (ms) **50403,436** sensor **ball**

t0 (s) **36** amplitude **727.769**

last frozen Previous Next

Trigger **active**

current rate (Hz) **10.3478**

global rate (Hz) **10.122**

Refresh clock (s) 1 2 3 4

Stack **44**

CPU

Debug

IP

trmt

evts

Close

baseline RMS



low amplitudes



high amplitudes



rise time



high amplitude signal



Noise



07:41:0

07:41:1

07:41:1

07:41:1

07:41:1

(OpiumR

07:41:2

07:41:2

07:41:3

07:41:3

(OpiumR

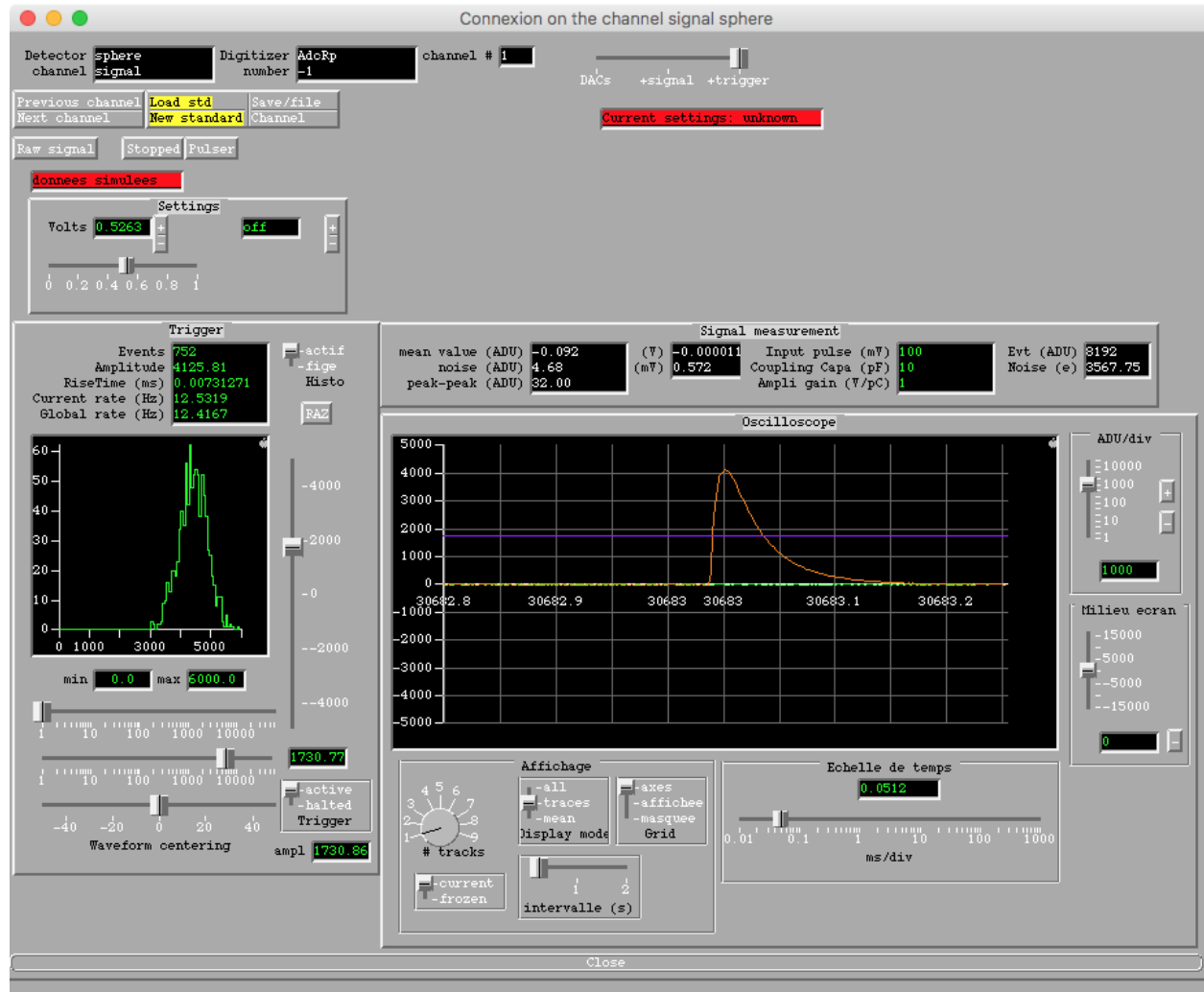
07:41:4

07:41:4

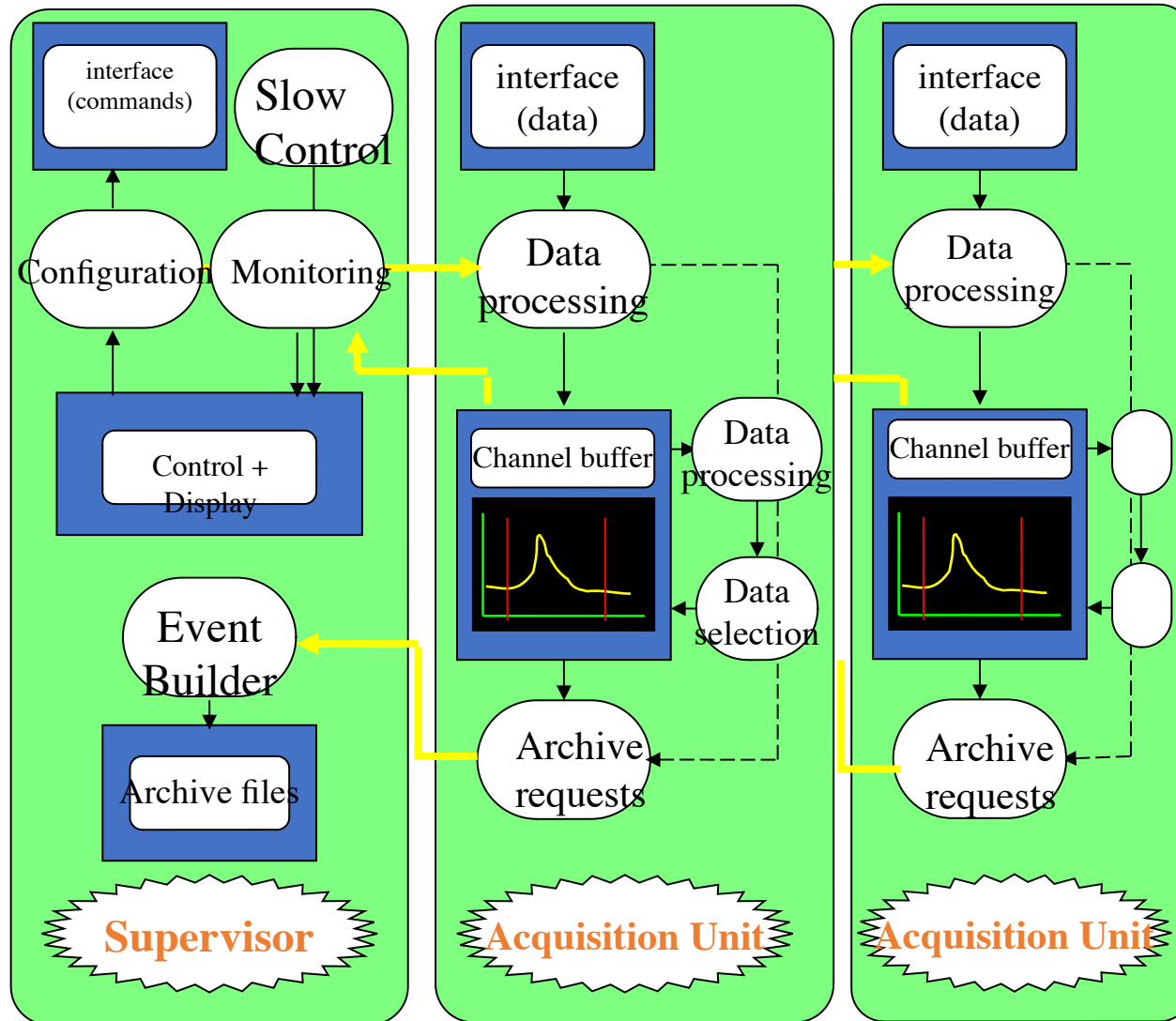
07:41:50/ Delai entre lectures anormal: 11496 us, niveau piles max pre

(OpiumRunGraph) KeyBoard en (0, 0) + (0, 0)

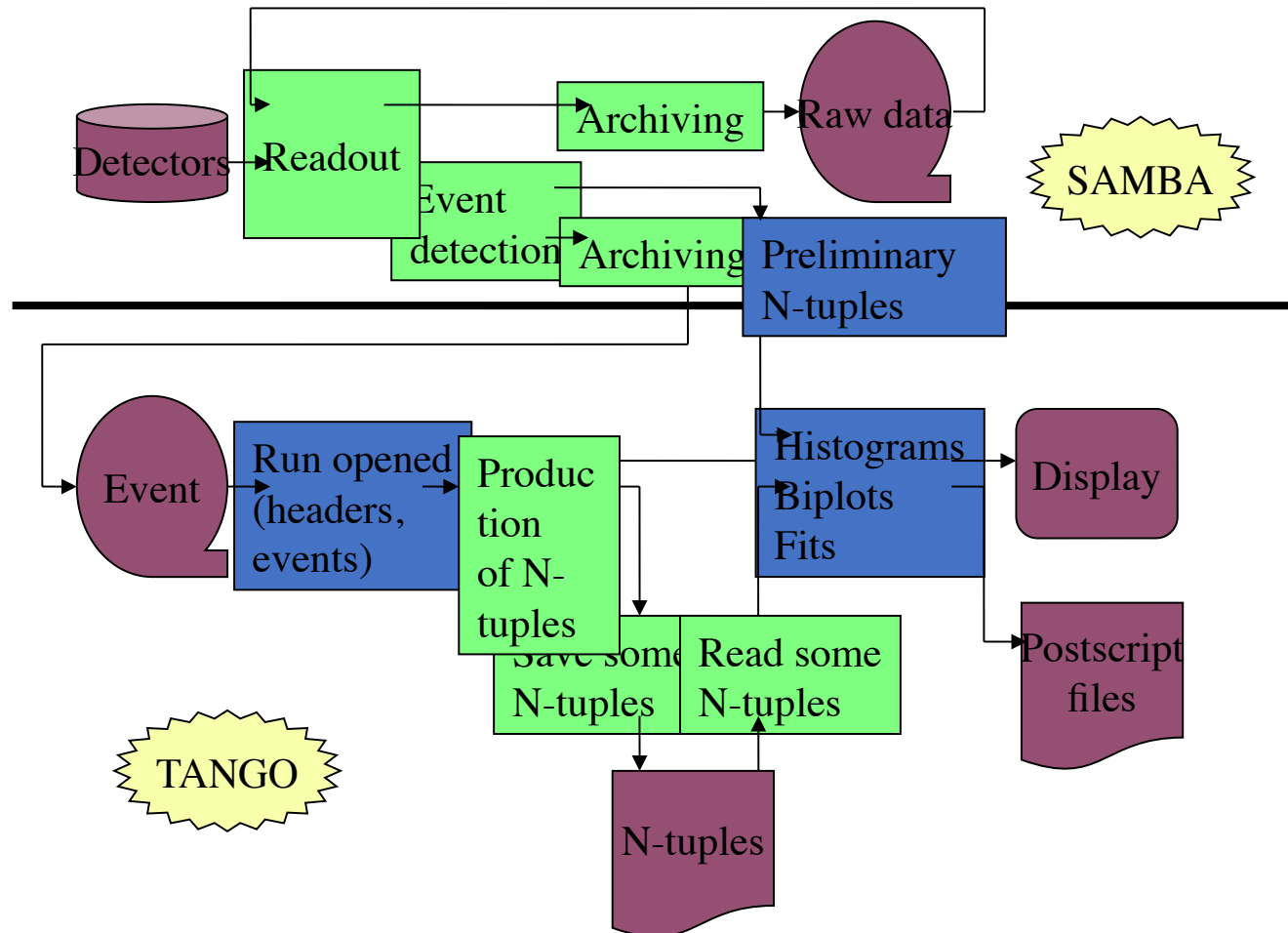
To aid with debugging, a live oscilloscope function is provided so you can monitor the channel inputs that SAMBA is seeing. In addition, you can check event triggering and definitions as well



For particularly large experimental setups, the SAMBA DAQ system can be split across multiple machines

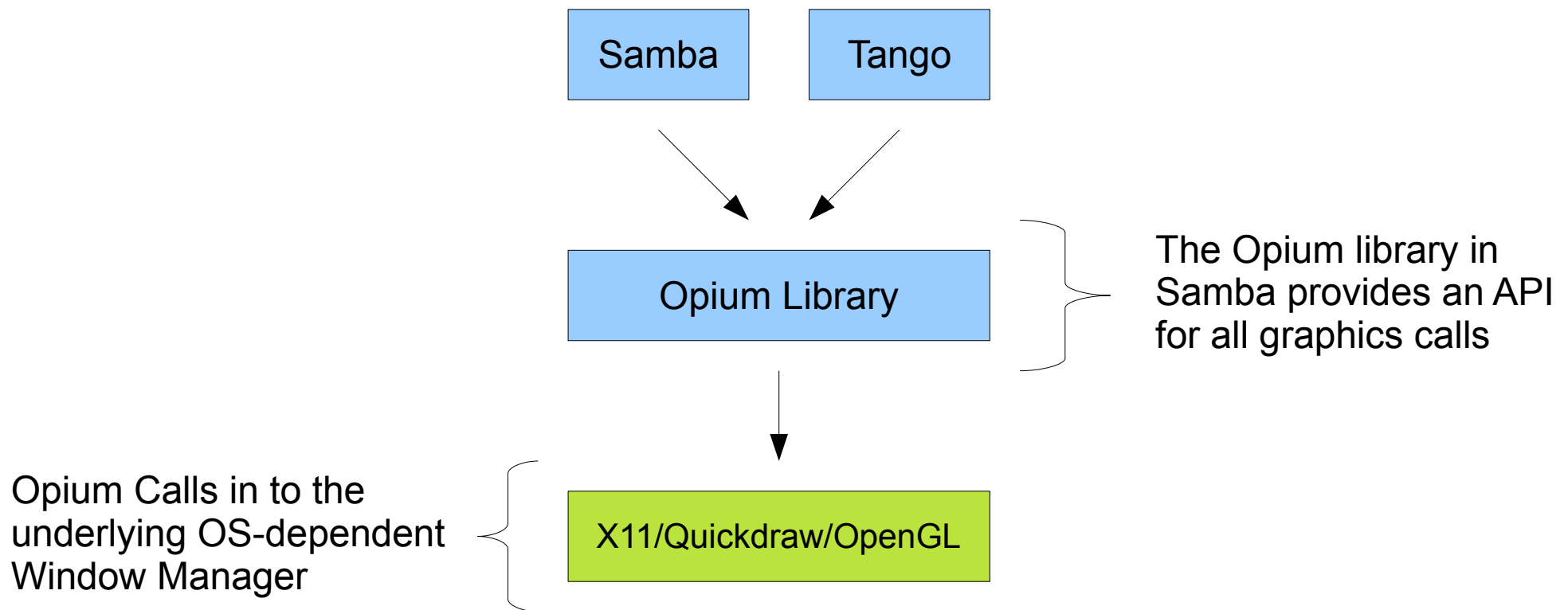


In addition to the DAQ part of SAMBA, there is also a data analysis part based on the same code base to aid with fast turnaround post-processing and Data quality checks



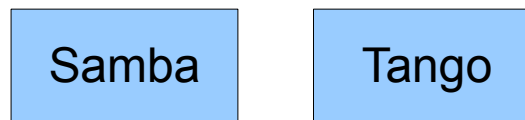
In the last 6-12 months, we at Birmingham started helping with ongoing SAMBA development, the main focus of which has been the GUI

The GUI has become increasingly difficult to support due to platform and WM dependant code being all that was available when the code was initially designed



Switching the GUI elements to a cross platform library (wxWidgets in our case) means that SAMBA is now somewhat insulated from changes to the underlying OS and libraries

This also had the added benefit of allowing it to be compiled and run on Linux (rather than just Apple) and possibly even Windows in the future



The Opium library in Samba provides an API for all graphics calls



Opium will call into WxWidgets which will then call the appropriate OS API





SAMBA is already very feature rich. However, there are a few areas where development is ongoing:

- Better integrated Slow control using externally defined JSON packets ●**
- Generic IP control for ADC boards ●**
- HTTP Access and communication ●**
- Easier distribution and configuration for new users ●**
- Improved Documentation ●**