TLU presentation

Thomas JACQUES

BUT - Bachelor of Technology

CNRS apprentice – 2 nd Year

IPHC laboratory

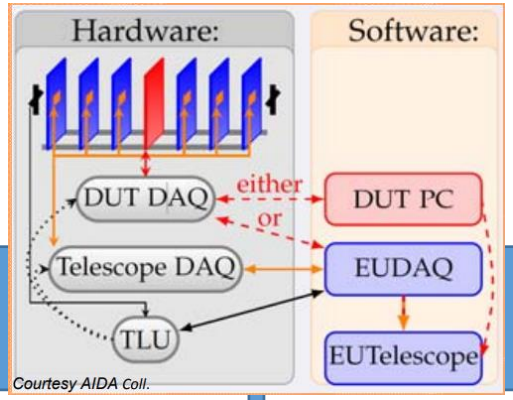
T.JACQUES
Apprentice
BUT1 GEII
Dev Test Benches
HW, SW

Contents

- I) What is EUDET ?
  - Contexte presentation : -FP6- EUDET - Detector R&D towards the international Linear Collider
  - EUDAQ : a generic data acquisition software framework
  - EUDET telescope
- II)Explanations of the TLU – Trigger Logic Unit
  - *The goal of the TLU*
    - *Examples*
  - The different TLU options
  - The different TLU modes
- III)Presentation of the Project-DEMO
  - Goals of the Demo
  - Project-demo overview
  - Results
- IV) Conclusion

Context presentation

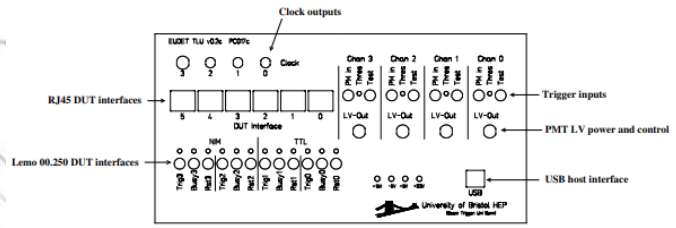
- The project goal was to create a coordinated European effort towards R&D for the next generation of large-scale particle detectors
- This project has developed infrastructures to simplify DUT integration in beam Telescopes ➔ Reduces development time



Hardware part

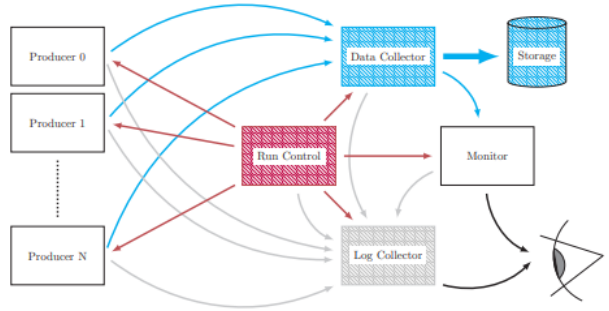
- Eudet telescope

- TLU : Trigger/busy logic



Software part

- Centralised but distributed core components that communicate with so-called Producers via a custom TCP/IPbased protocol.
- EUDAQ controls all components globally, handles the data flow centrally and synchronises and records the data streams



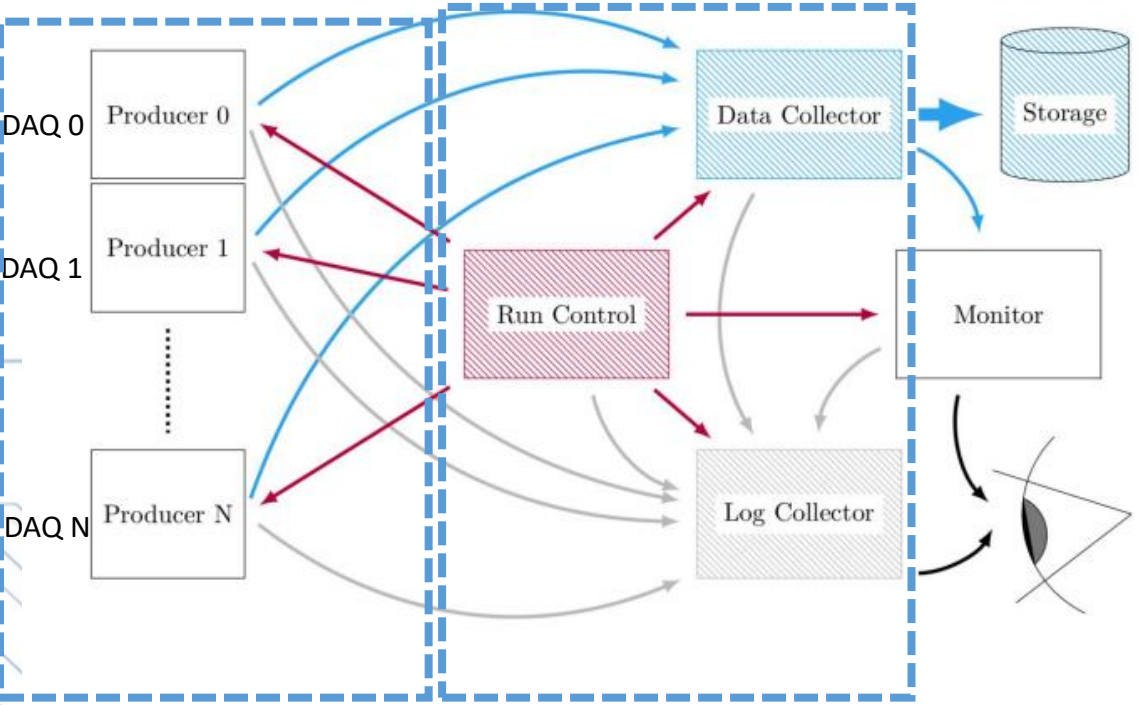
What is EUDAQ ?

- A data acquisition software framework for common beam telescopes

Software part

DAQ + producer softwares

One central PC



Run Control provides an interface to the user for controlling the DAQ systems :

- Configure the components (DAQ of each sensors – DUT/Ref)
- Sends commands to start and stop a run*
- Monitors the state of all processes

Each process of the DAQ that produces data is controlled by a Producer :

- Producer class are providing a standardized interface
  - => defines a way to receive commands, sending datas and log messages
- Sends special beginning/end of run events
- Should send one event/trigger
- If it has no Data for an event => sends an empty event

Log messages from all processes are collected centrally in the Log Collector, using the dedicated log channel

- The user is able to monitor all distributed processes for any unexpected warnings or errors

The Data Collector receives data as EUDAQ Event objects from all data-generating processes over the data channel

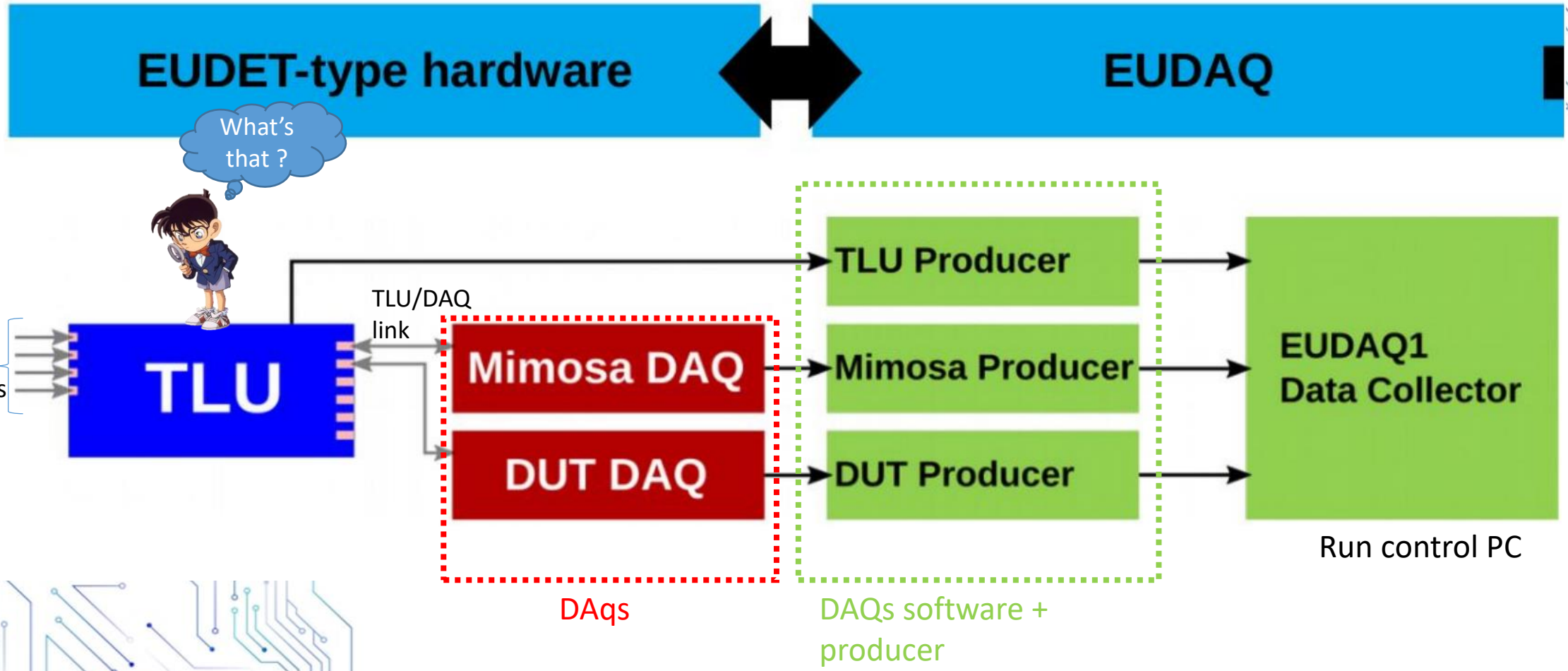
- Orders and synchronizes the events from the producers

**Run : indicates a continuous period of data taking, the runs are divided into events*

What is EUDAQ ?

Software part

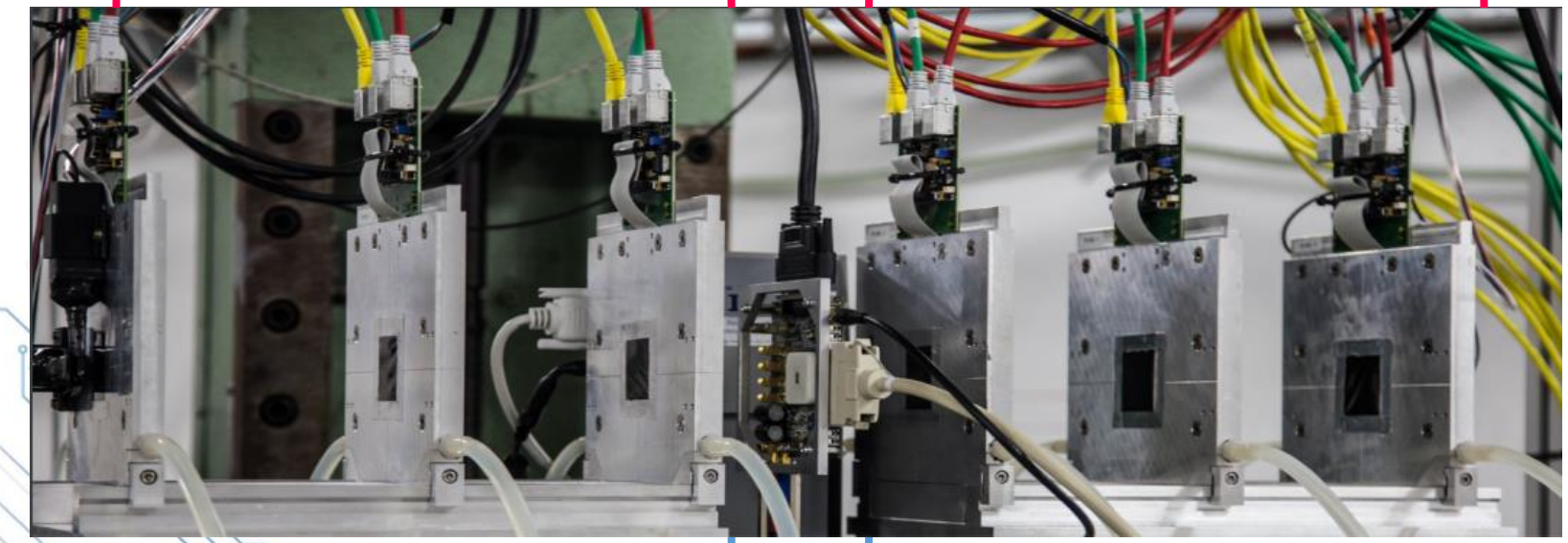
Example of the system with two DAQs : REF ➔Mimosa 26 and DUT➔DUT



What is EUDET ?

Hardware Part

Reference DAQ



DUT DAQ

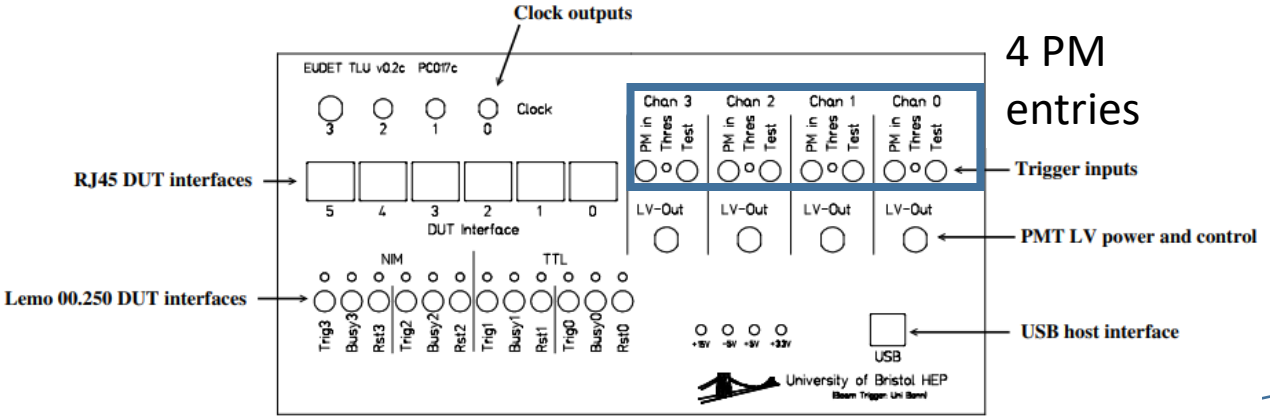
Eudet telescope :

- high-precision pixel beam telescope use to characterize pixel sensors
- versatile tool for a wide variety of devices
  - ➔ easy integration strategy for devices under test (DUT) and its DAQ
  - **➔the DAQs are synchronised thanks to a trigger-busy protocol**

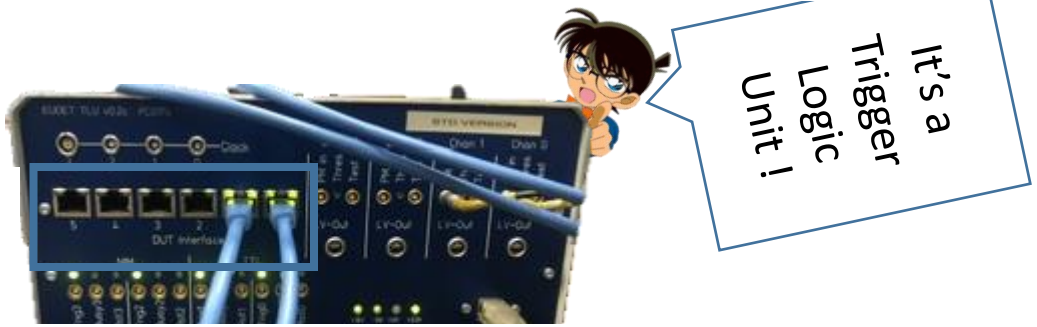
How to synchronise those DAQs?

Explanations of the TLU

Goal of the TLU :



4 PM entries



It's a Trigger Logic Unit !

6 TLU/DAQ link
(using RJ45 connectors)

Simple Handshake

The Goal of the TLU is to 'synchronise ' the different DAQs to do that it uses a trigger-busy logic.it works as follows :

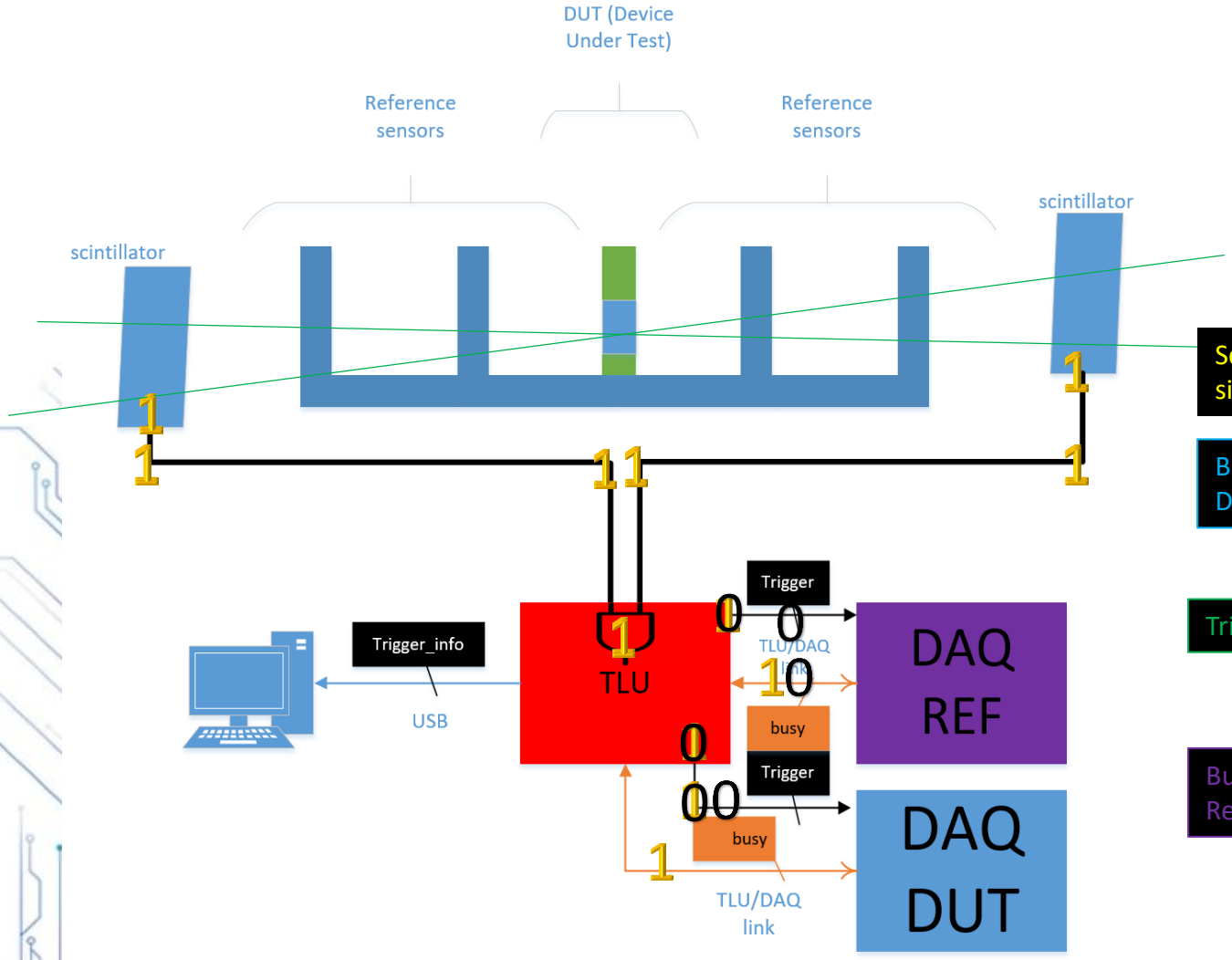
- The right combination on the 4 Pm (Photomultiplier) entries is received
- The TLU sends a TRIGGER signals to the selected DAQs entries among the 6 available.
- In response to the Trigger signal, the DAQs are putting there 'BUSY' line to 1
- The TLU sends the trigger line to 0 and won't send any new Trigger signal before every DAQs' 'busy' lines are set back to 0

- Along the DAQs synchronisation, the TLU is also sending Trigger info to the run control PC (timestamping) so that Datas can always be linked between the two DAQs

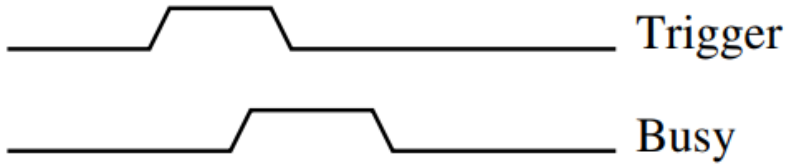
Goal and TLU explanations

Example : Use of 2 PM entries and 2 DAQS, ref and DUT(slower)

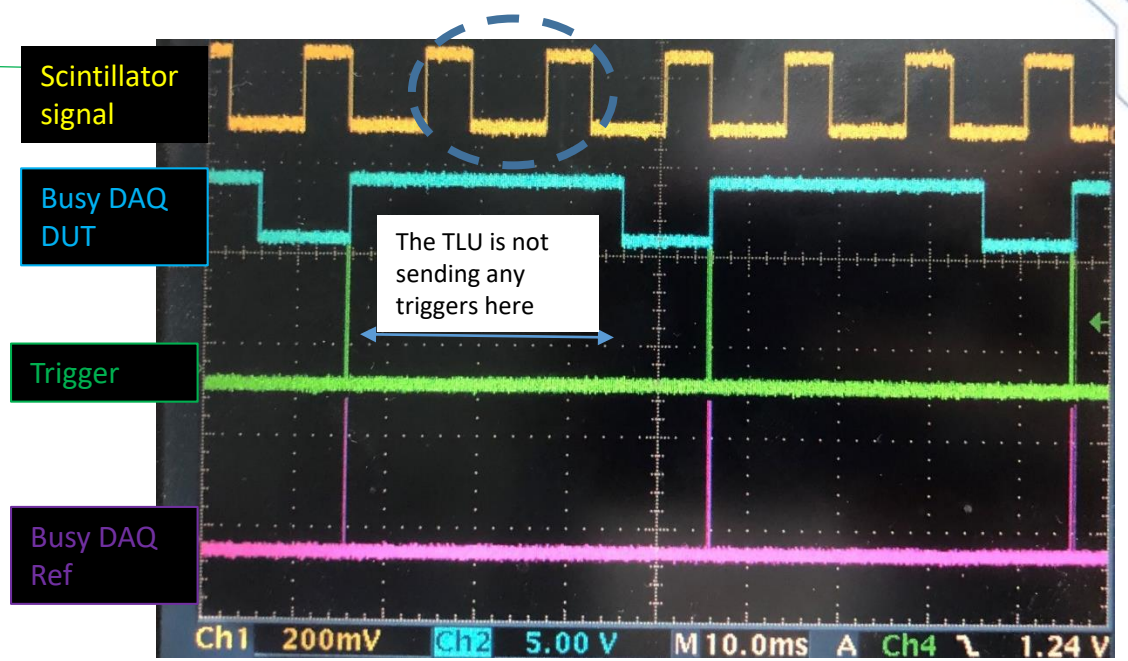
The TLU is set-up to wait for a 'coincidence' on the 2 PM entries to assert TRIGGER line.



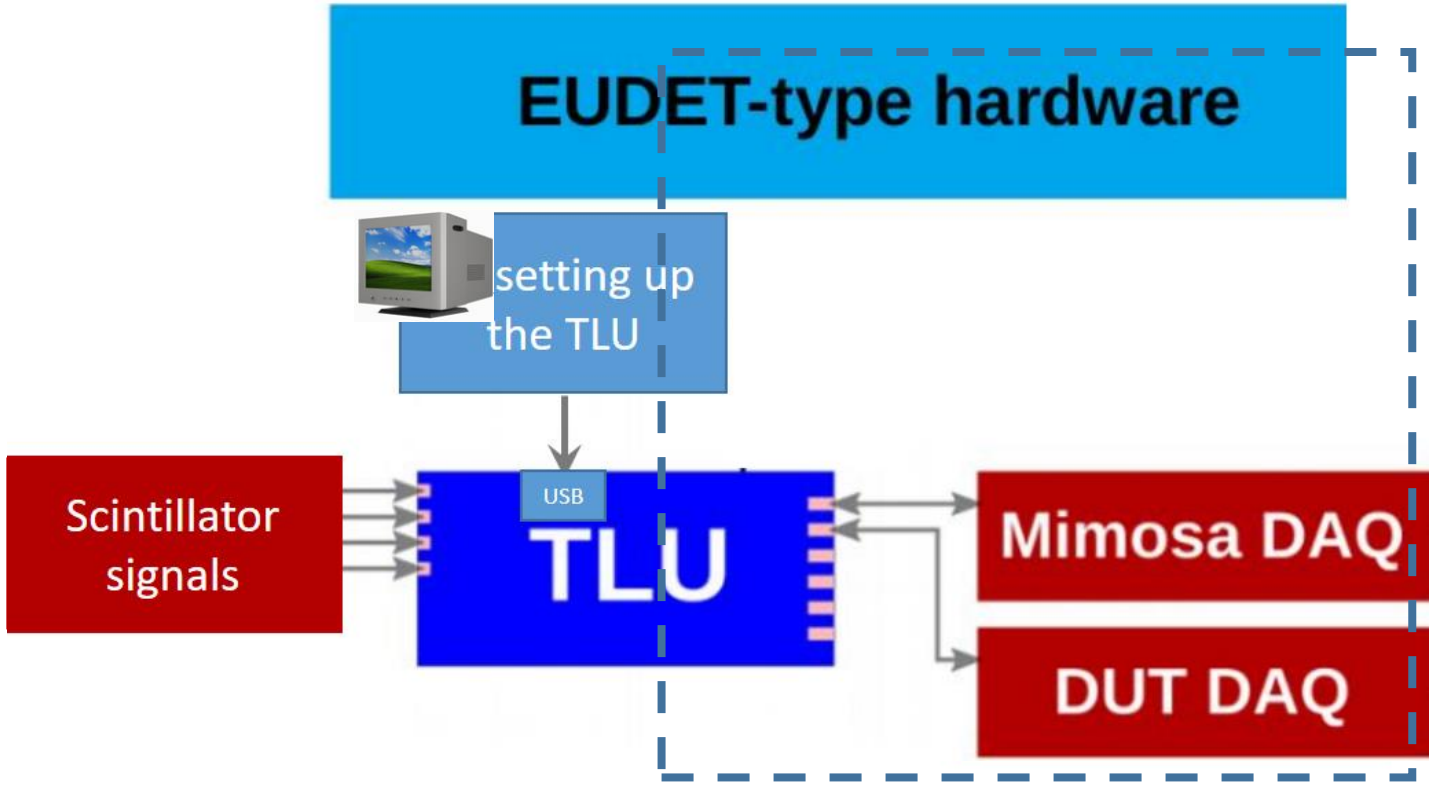
Simple Handshake



These Scintillator signals are seen by the TLU but the TLU isn't sending them as long as a BUSY line is at a HIGH state



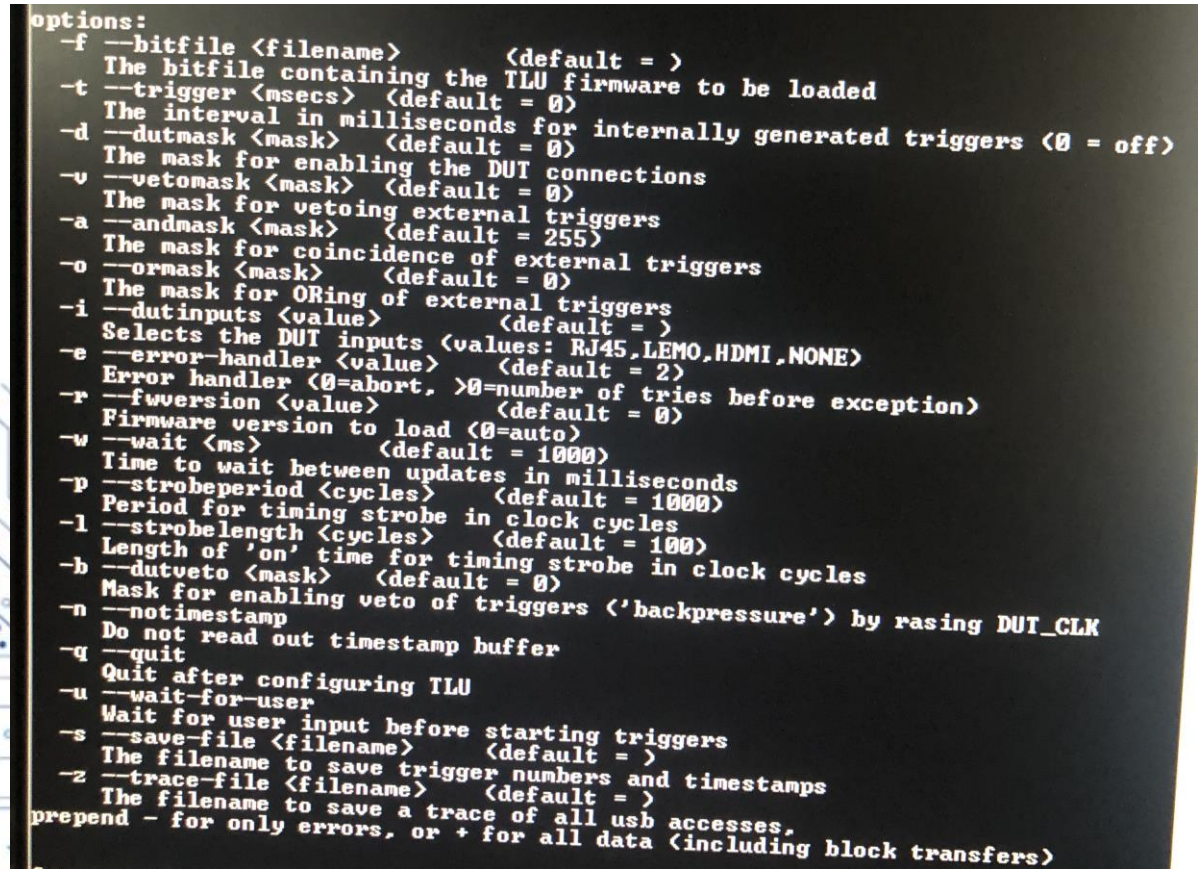
Now focusing on the options of the TLU that can be set-up thanks to a command line software on windows 32
➔ making a batch is advised



The following information are true for the TLU V0.2C (EUDET TLU), there's a more recent version: the TLU V1E_F, (TLU AIDA) which has more modes/options. A new one is being designed at the moment the TLU AIDA Innova

Explanations of the TLU

TLU's options

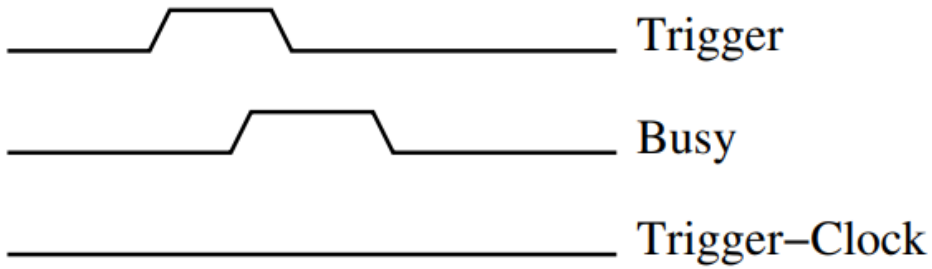


This menu is accessible by typing TLUcontrol.exe –help in a windows 32 command prompt (as the software is only available on windows 32)

The TLU has many options here are some you have to set-up before using the TLU:
-d : defines the DAQ-ethernet port you're using /!\ the decimal value entered is converted into binary code
-b : defines the DAQ-clocks that can veto the Trigger sending. 255 = all entries
-a and –o : **/!\ the two registers are inverted so –o 255 is a coincidence between all PM entries**

Explanations of the TLU

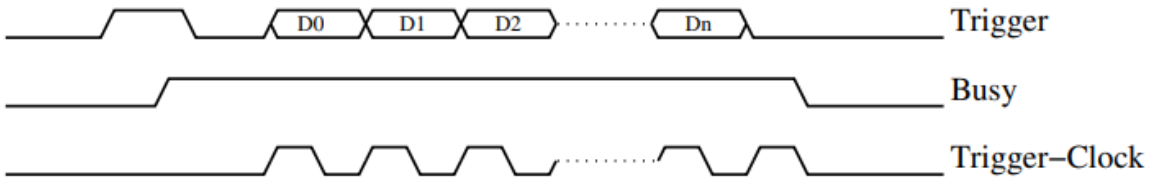
TLU's mode : simple data handshake



- TLU receives trigger from beam scintillators
- TLU asserts TRIGGER
- On receipt of TRIGGER going high, the detector asserts BUSY
- On receipt of BUSY from DUT, the TLU de-asserts TRIGGER
- On receipt of TRIGGER going low and the detector being ready to take more data, the DUT de-asserts BUSY

Explanations of the TLU

TLU's mode : Trigger Data Handshake



- The beginning is the same as in the simple handshake mode
- On receipt of BUSY from DUT, the TLU de-asserts TRIGGER
- On receipt of BUSY going high, TLU de-asserts TRIGGER and switches the TRIGGER line to the output of a shift register holding the trigger number/data.
- The DUT clocks data out of the shift register by toggling TRIGGER CLOCK. Data changes on the rising edge of TRIGGER CLOCK The 15 least significant bits of the trigger data are shifted out .
- The DUT should issue 16 clock pulses which will clock out the bottom 15-bits of the trigger number and put the trigger line back to 0.

TLU's mode : No Handshake

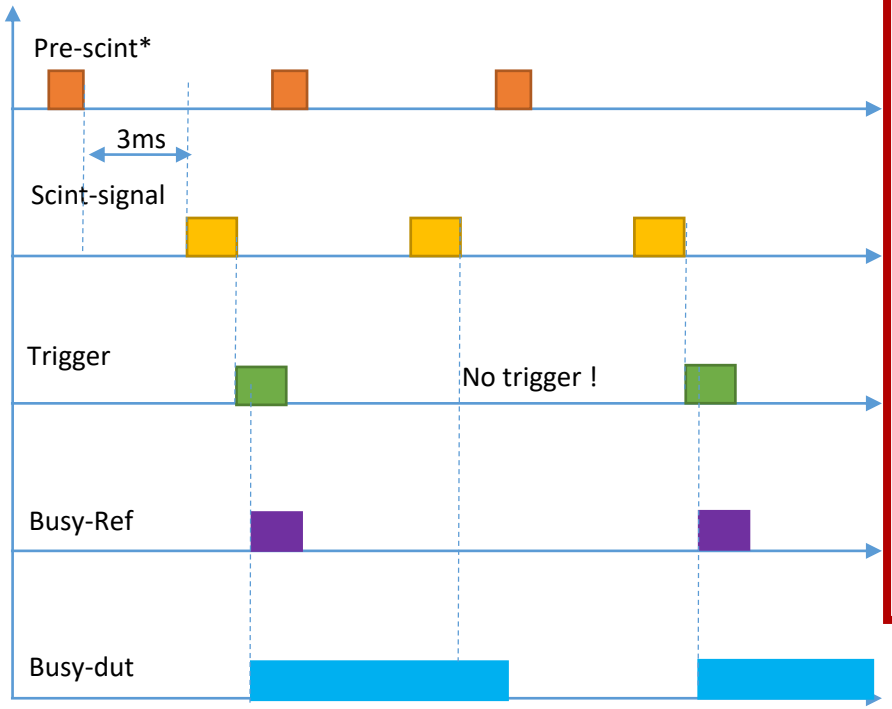
- In this mode the TLU issues a fixed-length pulse on the trigger line

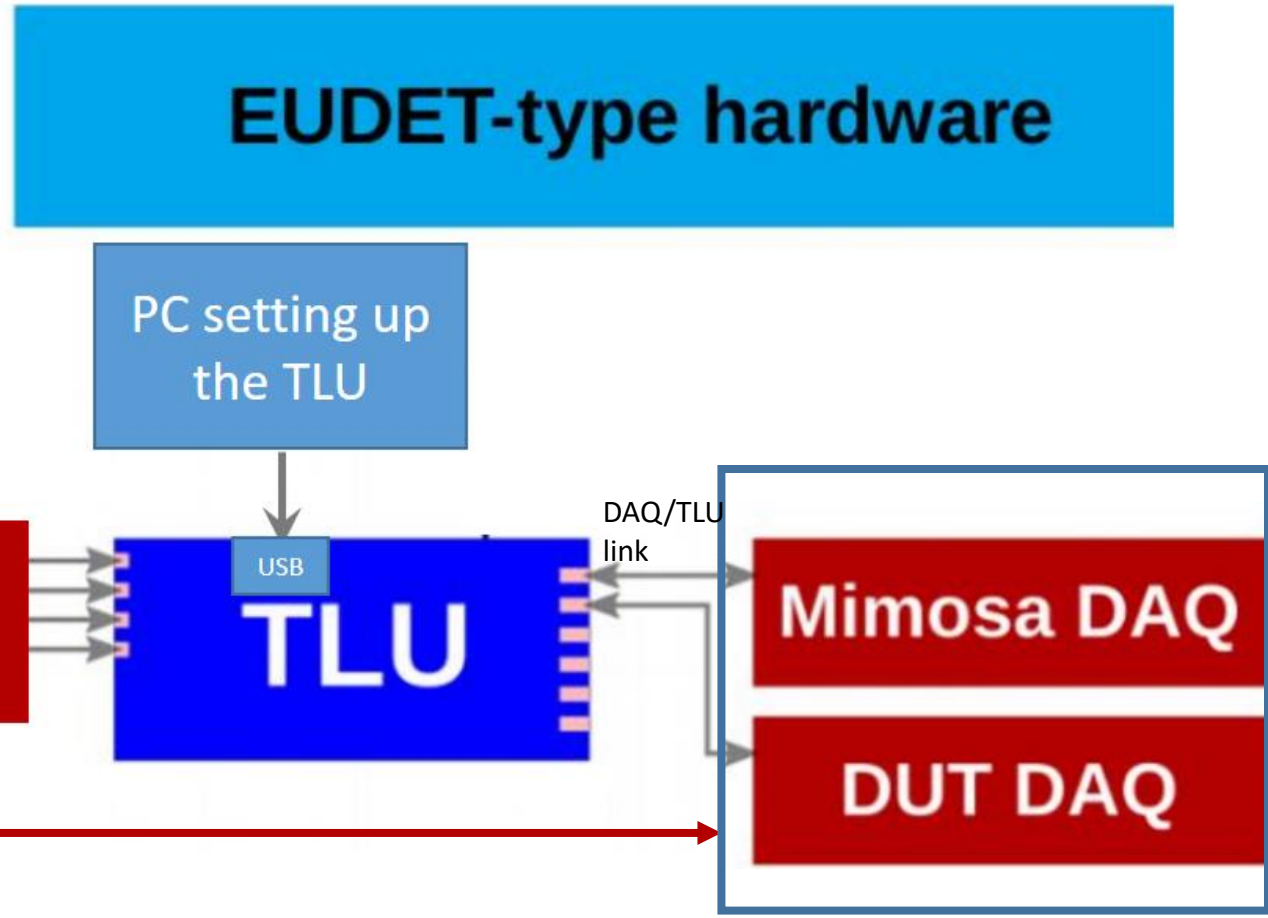
Presentation of the Project-DEMO

Project Goal :

The goal is to validate the working principle of the TLU to do that we need to :

- Emulate a scintillator trigger emulator
- Emulate 2 DAQs : Ref and DUT1 (same type of signals, different pinout)

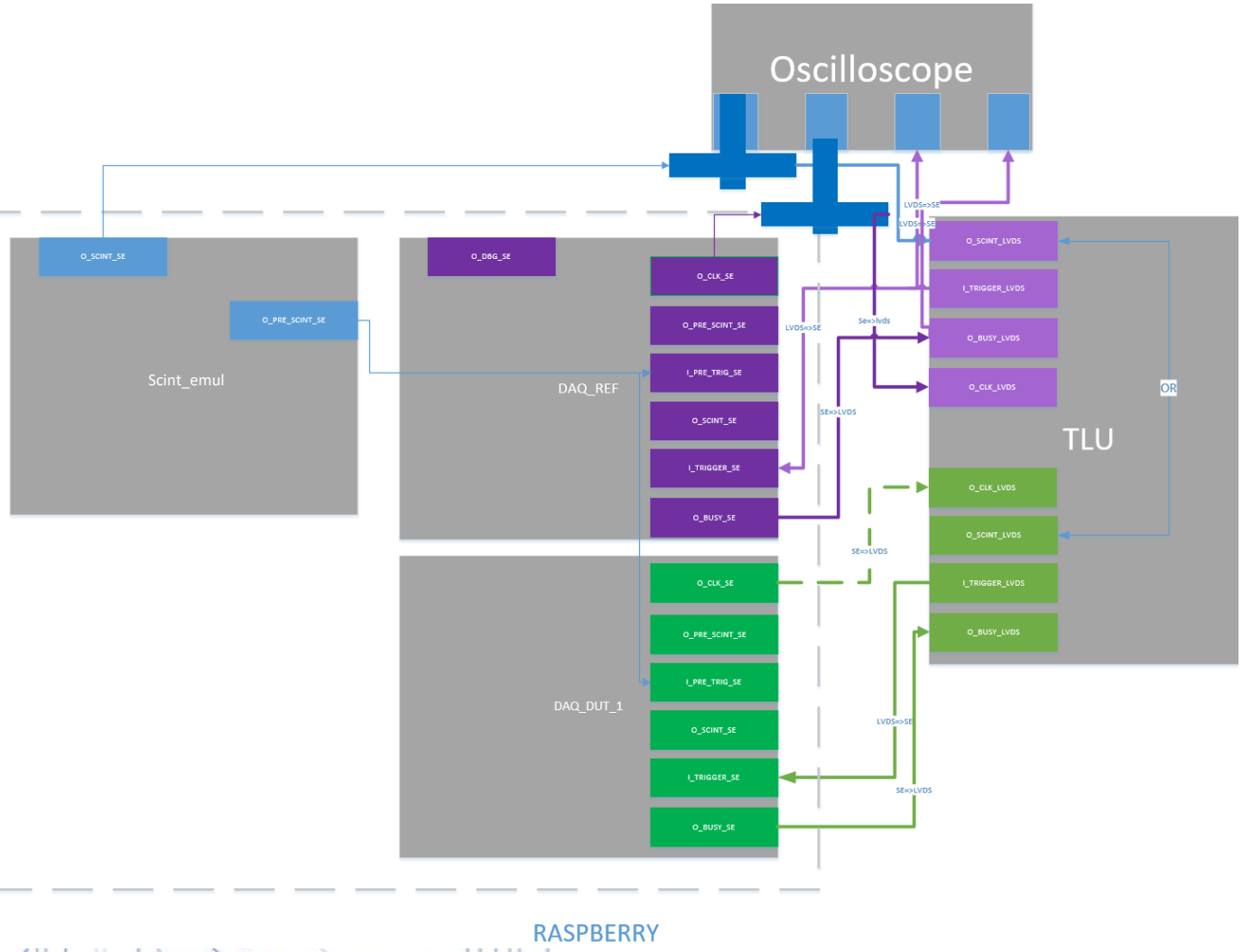




*The pre_scint signal doesn't exists in real test beams, we implanted it to be sure to count the real number of Trigger sent on each DAQ .

Presentation of the Project-DEMO

Project Goal



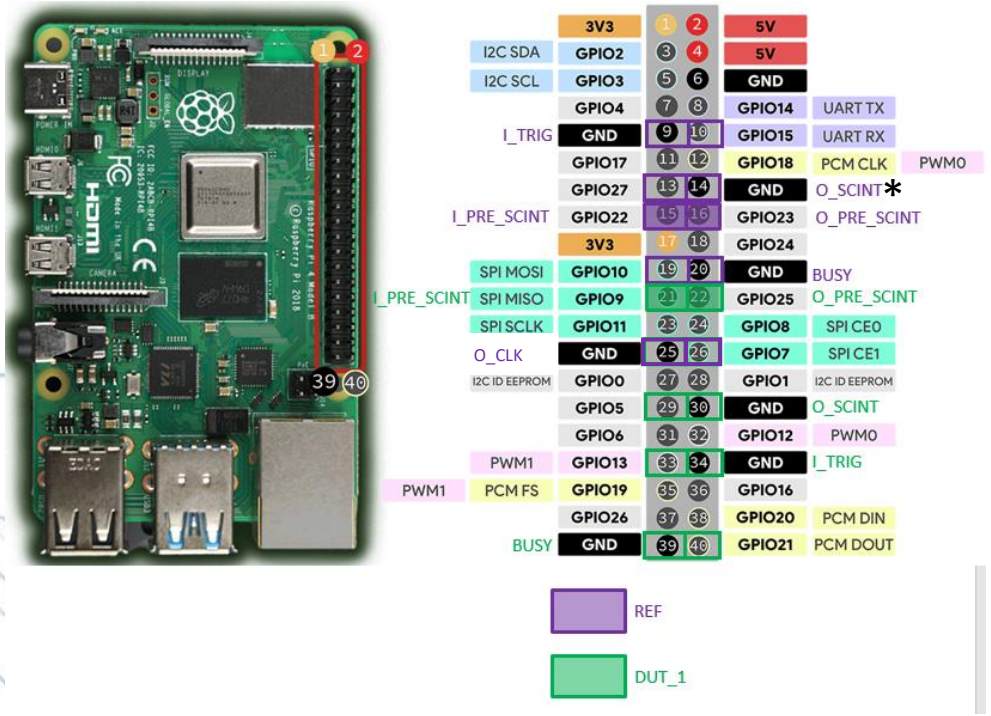
Goal :

- Emulate 3 different component thanks to a Raspberry Pi 4 model B :
  - A scintillator trigger emulator
  - 2 DAQs : Ref and DUT1 (same type of signals, different pinout)
- The DAQs must have these options :
  - Initialisation (initialise the I/O of the Rspy)
  - Reset : Reset the different counters
  - Emulation of Scintillator signals
  - Run configuration
    - Acquire N-Events/infinite number of events
    - Displays Status of the run
    - Run number chosen by the user
  - Start/Stop the Run
  - DAQ_Delay : emulates a DAQ slower than the other
  - Quit : Stops the program

Presentation of the Project-DEMO

Hardware part of the project

The pinout is the following :

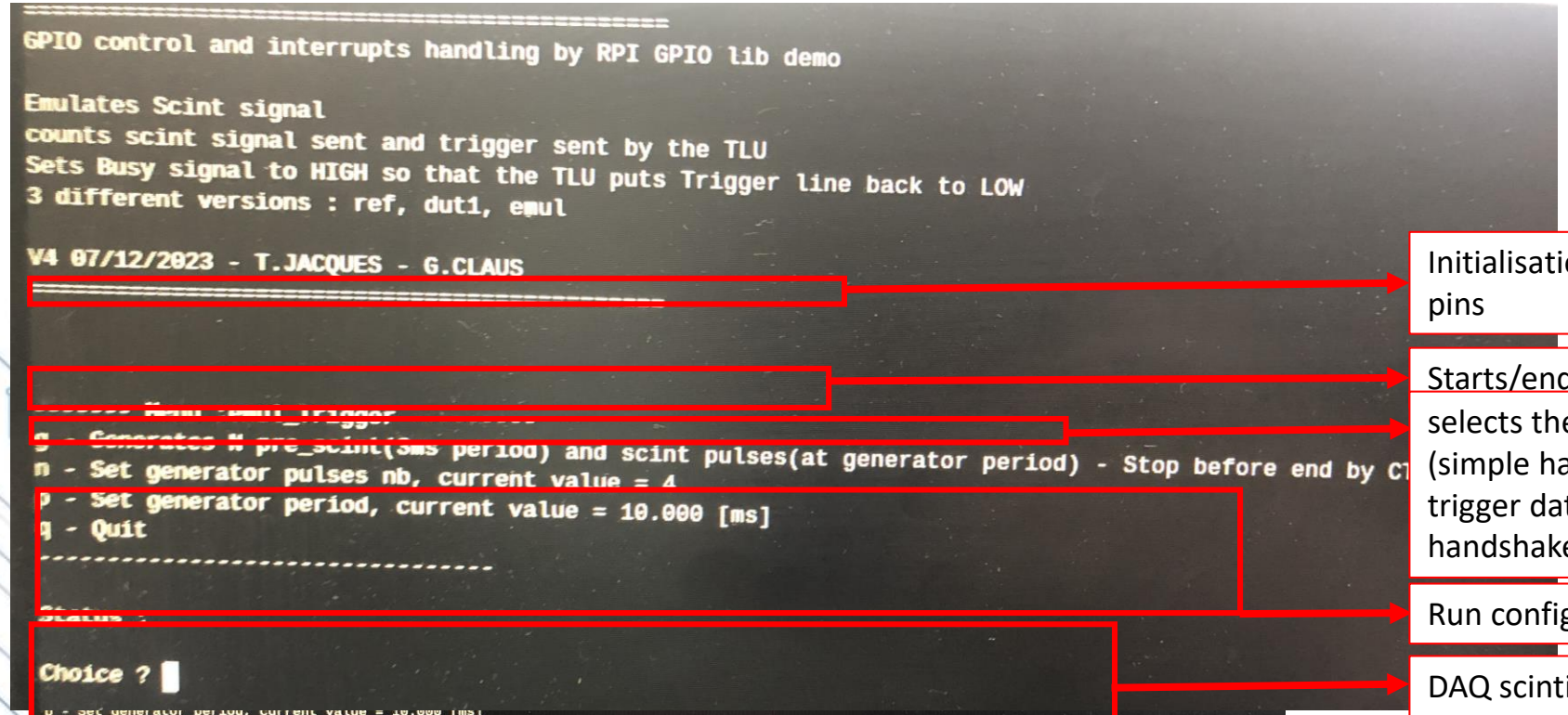


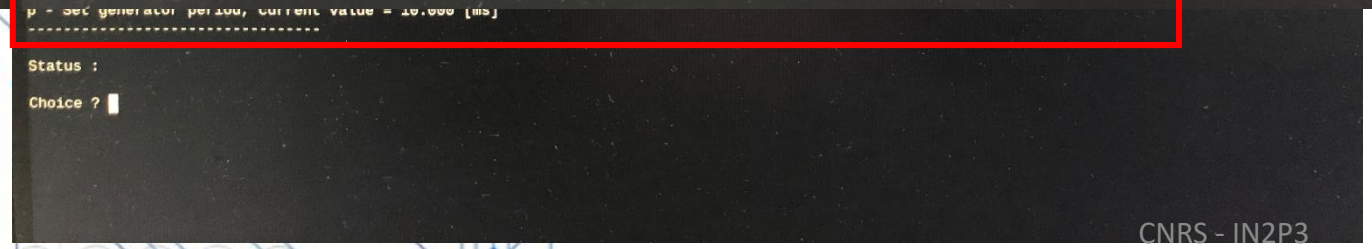
*the same pins are used for the scintillators triggers of the ref-DAQ and the scintillator emulator

Output signals	Input signals
• O_SCINT : scintillator emulation signals • O_PRE_SCINT : pre_scintillator emulation signals, doesn't exist in reality but thanks to this signal we were sure that the count of pulses we were sending to the TLU were right • BUSY : emulation of the busy signal • O_CLK : emulation of the TRIGGER_CLOCK signal that enables the TLU to clock_out the 15 bits of the trigger number register	I_TRIG : Trigger signal comming from the TLU I_PRE_TRIG : Pre-Trigger signal coming from the RSPY, as the 'O_PRE_SCINT' signal, doesn't exist in reality

Presentation of the Project-DEMO

Software part of the project





Initialisations of the pins

Starts/ends the run

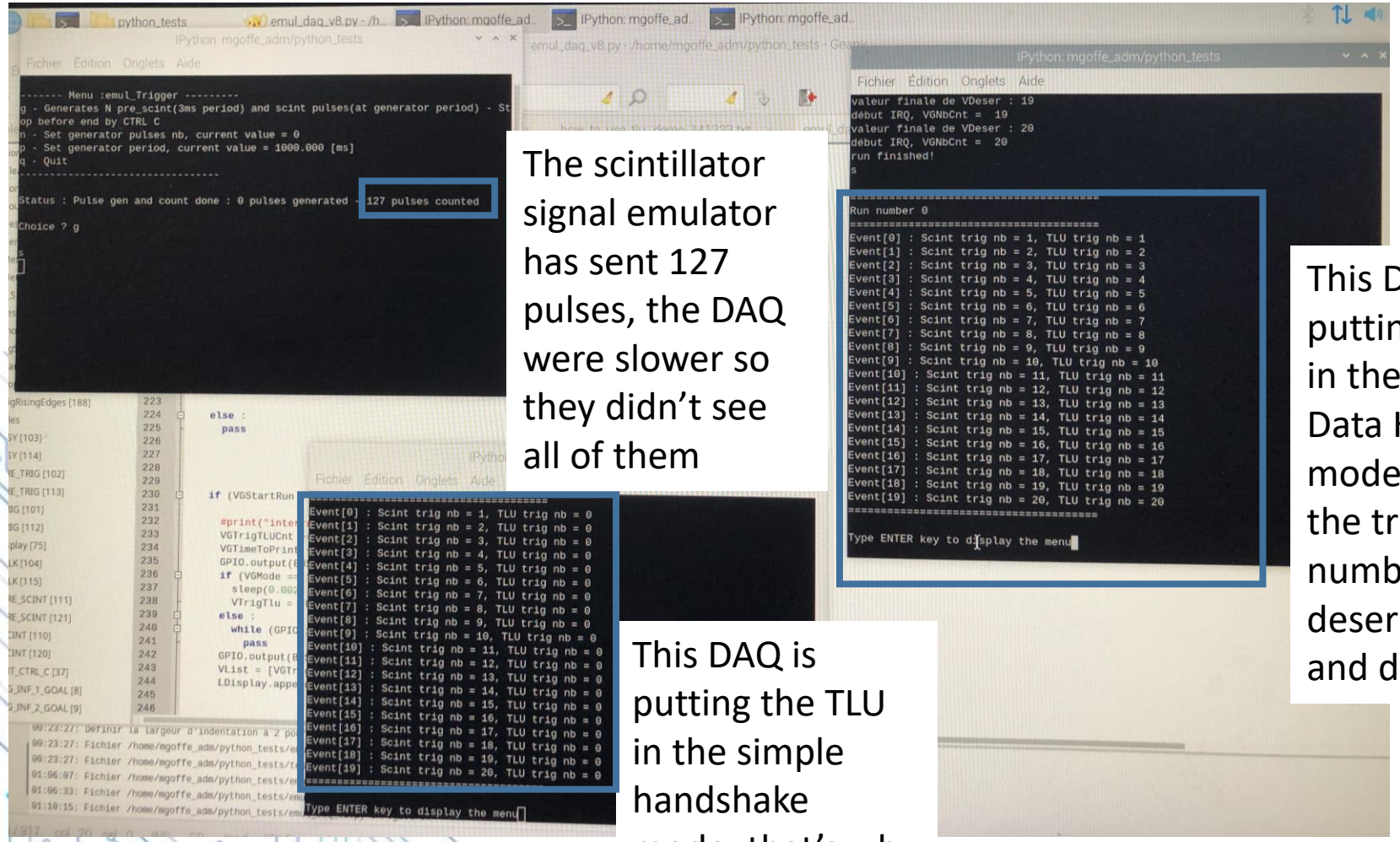
selects the mode (simple handshake of trigger data handshake

Run configuration

DAQ scintillator signals emulation

Presentation of the Project-DEMO

Results



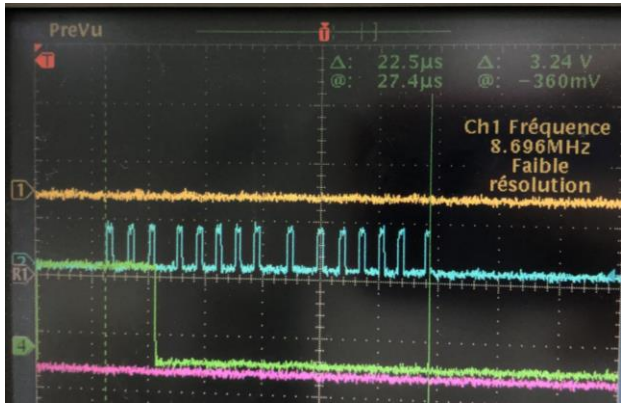
The scintillator signal emulator has sent 127 pulses, the DAQ were slower so they didn't see all of them

This DAQ is putting the TLU in the simple handshake mode, that's why « TLU trig nb » stays at 0

This DAQ is putting the TLU in the Trigger Data Handshake mode, it receives the trigger number, deserialises it and displays it.

Conclusion

- The first project objectives have been met, it is possible to emulate two DAQs and a scintillator emulator at the same time using one raspberry. Two of the three TLU's modes (simple and trigger data handshake) are working (no access to the software so we can't try the third mode at the moment)
- In simple handshake mode we can at least emulate a 300Hz clock-scintillator signal, we can certainly go faster but we didn't had the time to test it.
- In trigger Data Handshake mode we can only have a 54Hz clock-scintillator signal at best (some glitches may happen, it is advised to set a tinier clock frequency). Without delays the total duration of the 16 clock pulses is about 22 to 25 µS at best. The signal has not a constant 50% duty cycle but when it is the case the period is 1.40µs long.



- The next step is to implement EUDAQ in the demo, once this is done it'll be time to use real DAQs to test the TLU in conditions near to test beams condition