DAQ Stack

Coincidence Processor

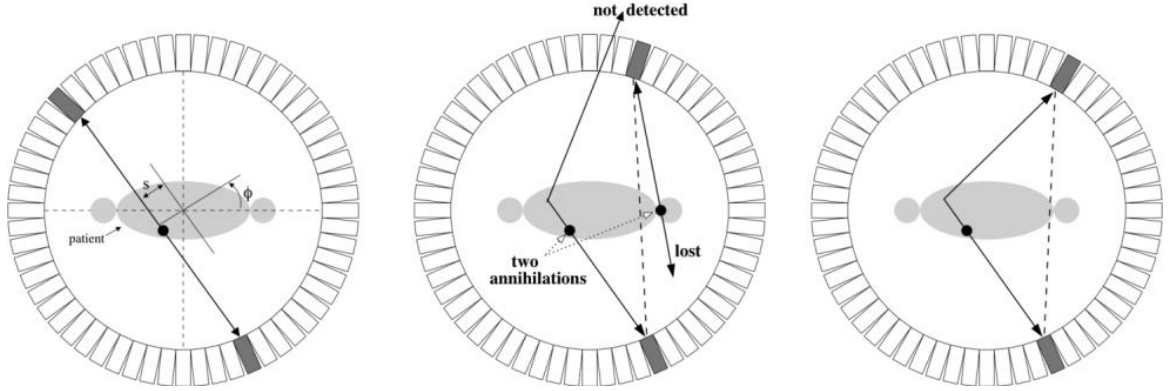
For XIA Pixie16 Data Acquisition System

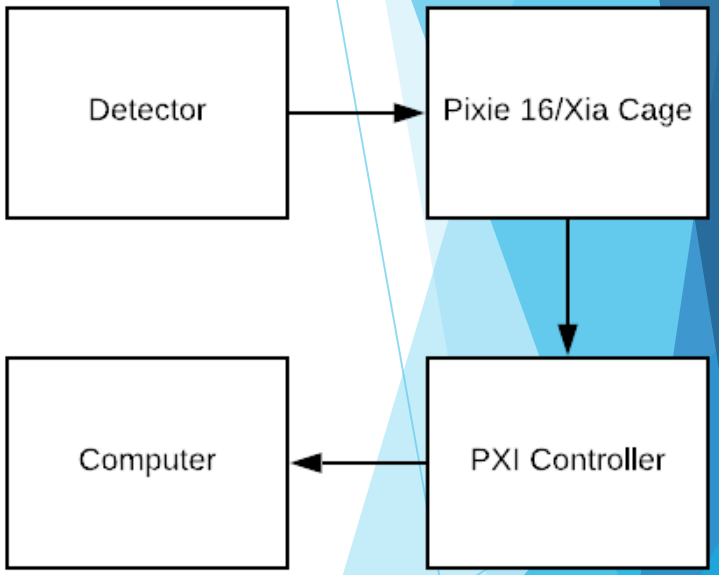
Kyle Leon Jordaan 3538638
Term4

Supervisor : Dr Michael Norman (Computer Science)
Co-supervisor : Prof Nico Orce (Physics)
Mentor: Dr Kushal Kapoor (Physics)

DAQ System

- A Gama ray detector has been bought by the MANDELAB in the physics department.
- This detector has been setup and development is underway to acquire the data coming from this detector setup as well as processing it for future experiments by the scientists.
- This system will ultimately be used in the development of a PET scanner. For this reason, the system is needed to identify coincidence gamma ray data.





Testing Background

- Ensures the DAQ stack Platform conforms to the design specification
  - Verified by means of user testing
- Helps ensure the stability of the platform

Design Specification

- Uses PAASS-LC for data Acquisition and processing
- Generates a time spectrum for the purpose of time calibration
- Threshold time calibrated events to find events in coincidence
- Generate the energy spectrum of the coincidence events
- All analysis occurs within a user-friendly web interface

Testing Strategies

- User testing
  - PhD students who use data acquisition as part of their thesis
  - Provide feedback on the application, if it abides by the requirements and future improvements
- Integration testing
  - Occurs at an API level
  - Ensures the features of the API and data Processing are working as expected
- Stress testing
  - Ensures that the system will not fail under abnormal conditions

Test design

- User testing
  - Users are given access to DAQ stack
  - The users process an experiment and download the results
  - The users comment on if the User requirements have been followed
  - Feedback is given as to what could be improved
- Integration testing
  - HTTP requests are made to the API using a Python client
  - Responses and side effects are monitored by the Python client
  - Alerts the developer if an error occurs

Test design

- Stress Testing
  - A virtual machine with limited system resources is used to simulate an abnormal situation
  - Large experiment files are uploaded to the system
  - Processing duration is recorded
  - System crashes and Web UI crashes are recorded

Test Cases

- User testing
  - Users are provided data files for simplicity
  - Users are instructed on how to write the configuration file
- Integration testing
  - An experiment is processed using the web UI.
    - If the results are satisfactory the response is recorded and stored for the Python test client
  - Also the results of the history and details screens for 4 predetermined experiments are recorded and used for validation
- Stress testing
  - Limits are set on the amount of ram and storage in a virtual machine

Test report

- User testing
  - Users provided positive feedback on the usability experience of the platform
  - They felt the requirements were met
  - They suggested future improvements of
    - Improving UI style
    - The ability to record data
    - User authentication
    - Extend processor code to use multiple detectors as apposed to simply 2

Test report

- Integration testing
  - Stability issues arose when supplying nonstandard data
    - Incorrect type formatting or broken XML config files
    - This was fixed by blocking nonstandard requests and validating config files
- Stress testing
  - Various performance issues arose during the stress testing phase.
  - Files larger than the available storage capacity caused the system to crash.
    - File sizes are now limited to 1 Gb smaller than the available capacity
  - Large amounts of graphs on the history tab of the UI cause the front end to crash
    - Nictitating the need for pagination of experiments

Conclusion

- During the development of the DAQ stack platform an issue was discovered in the PAASS-LC platform which caused it to crash when supplied with our data. This was brought to the attention of the PAASS-LC developers and has since been rectified
- DAQ STACK is a platform for the simple processing of experiment data.
- It can be easily extended upon given its modern language use and well-structured code base.
- Integration Testing provides future developers with a means of discovering faults they may have introduced into the code base