Extraction of Time and energy data from a digital pulse processor

K. Jordaan
University of the Western Cape
Private Bag X17
South Africa
3538638@myuwc.ac.za

ABSTRACT

A brief overview of the importance of time-energy measurements in the field of particle physics. And the uncertainty given by the Heisenberg uncertainty principle to the measurements being made. A discussion about the software which will be developed to allow for the accurate extraction of time and energy data as well as the analysis tools which will be developed to process this data. Finally resulting in an experiment at iThemba Labs which this software will be crucial in the verification of the detector process being developed.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Data Acquisition* → Analysis;

KEYWORDS

Digital Pulse Processors, Data Acquisition, Spectrum Analysis, digital partial detector processing.

1 INTRODUCTION

C.W Fabjan defines particle detectors as "Particle detectors are instruments used to measure the kinematic properties of particles and quanta" [1]. The kinematic properties are mass, position velocity and acceleration. These properties can be derived from the characteristics of detection events. Namely the energy of a particle and the time at which an event occurs.

Once an event has occurred it is registered in the Data Acquisition (DAQ) system and stored on a computer via an interface program. For this use case a software package called PAASS-LC [2] will be used as it provides both acquisition and analysis frameworks. However, this package is not designed for the specific use case of time and energy signal retrieval from multiple detectors simultaneously to register individual events. But rather it is designed in a general manner that allows it to be extended by a researcher to accommodate for their specific use case.

My Honors project will be the development of a tool using the PAASS-LC frameworks to retrieve time stamped energy events from the DAQ system. The data retrieved will allow for the time calibration of the particle detectors. During the decay of various radioactive isotopes, a gamma ray pair is produced which are then emitted in opposite directions. This time calibration is essential in the measurement of these decay events, as it allows for the measurement of the position of the decayed particle with respect to the Heisenberg uncertainty principle. This position with coupled with the energy of the emitted gamma rays can be used to determine the various other kinematic properties of the particle.

2 COMPUTATIONAL DETAILS

2.1 Time Calibration

To calibrate the detectors a radioactive source is placed a predetermined distance form either apposing detector. Then by measuring the precise time of arrival of the gamma ray emitted during a decay it is possible using newtons equations of motion to calibrate the detector in time with respect to these positions.

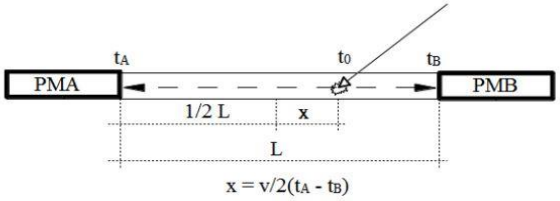

Figure 1: Diagram Depicting the calculation of the position of a radioactive point source in 1D space

If the time of an event at Detector A occurs at t_A and the event at Detector B occurs at t_B it is possible to determine the position of the particle between these detectors using Newtons Equations of motion.

This which gives the Equation displayed in Figure 1:

$$x = \frac{v}{2}\ (t_A - t_B) \qquad \text{Eq (1)}$$

2.2 Energy Calibration

The energy calibration is accomplished by using a radiation point source whose emitted energy spectrum is well defined.

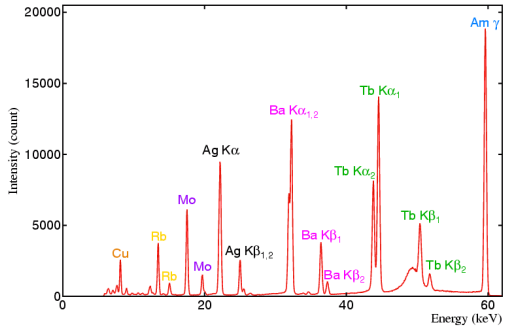

Figure 2: An Example Energy Spectrogram using a variety of radioactive sources [3]

Table 1: A table representing the energy levels of the various photopeak's in Figure 2.

	$K\alpha_2$	$K\alpha_1$	$K\beta_1$	$K\beta_2$
Cu	8.042 keV			
	(1.54178 Å)			
Rb	13.377 keV		14.963 keV	
	(0.92688 Å)		(0.82863 Å)	
Mo	17.446 keV		19.610 keV	
	(0.71069 Å)		(0.63225 Å)	
Ag	22.108 keV		24.946 keV	25.459 keV
	(0.56083 Å)		(0.49701 Å)	(0.48701 Å)
Ba	31.820 keV	32.197 keV	36.382 keV	37.261 keV
	(0.38965 Å)	(0.38509 Å)	(0.34079 Å)	(0.33275 Å)
Tb	43.745 keV	44.478 keV	50.399 keV	51.747 keV
	(0.28343 Å)	(0.27876 Å)	(0.24601 Å)	(0.23960 Å)

Figure 2 (for explanatory purposes) is a great example of a calibration spectrum. Using the values from [3] represented in Table 1 the energy in keV will need to be mapped to the channel number listed on the x-axis of Figure 2. This is accomplished by plotting the graph of channel number to energy. Then finding the line of best fit for this data. As the channel numbers n of a detector are approximately proportional to the energy E being measured the points on the graph will follow a quadratic relationship at high energies. The calibration step is reduced to the process of solving for the constants: a, b, c in the equation

$$E(n) = a + bn + cn^2 \qquad \text{Eq (2)}$$

This can be accomplished using the method of chai-square defined in [4].

$$x^2 = \sum_{n_i}^{n_f} \left(\frac{E(n) - Expected(n)}{\sqrt{Expected(n)}} \right) \qquad \text{Eq (3)}$$

Where n_i is the first channel number, n_f is the final channel number, E(n) is the energy predicted by Eq (2), Expected(n) is the expected energy at the channel number n.
Eq (3) is used by varying the constants a, b, and c until the value produced by chi-shared is the closest to 1, making this an optimization problem. Various optimization algorithms will be implemented and tested including Global Search, Multi Start, and Pattern search. They will be evaluated, and the optimal solution will be used in the project.

3 FEATURES AND METHOD

3.1 Tool To Be Built

The resulting system will be built upon the PAASS-LC tool. It will expand on both the acquisition and analytical frameworks provided by the tool.

- It will use PAASS-LC as a data recording program to interface with the DAQ.
- The program will then calibrate the energy spectrum incident on the detectors using the method of chi-squares in Eq (3).
- The subsequent events incident on the calibrated energy channel will then be recorded with its precise time stamp.
- Using this time stamp only the events occurring within a specific time frame threshold will be recorded.
- Using Eq (1) the precise position of the events will be determined, and the various other kinematic properties will be determined from the position.

Initially this system will be built as a processer within the PAASS-LC tool. Then a modern user interface will be developed for ease.

3.2 Development Cycle And Goals

Even though there is a clear and well-defined problem which needs to be developed, as with any software project there is still space for the requirement to change and as such an Agile approach towards the software engineering cycle will be followed. With regular standup meetings with my mentor as we discuss the changing nature of the project as we learn more about the capabilities of the PAASS-LC framework.
The major goal of this project is to provide scientists with an easy to use data acquisition and analysis system which they may use for coincidence experiments but may easily be extended to various other experiments as well.

4 USER REQUIREMENTS

4.1 Data Acquisition

This process is handled by poll2. A tool supplied by the PAASS-LC framework and will not be built by this project.

4.2 Data Analysis

Poll2 will acquire data and store it in in a ldf file. UTKscan a tool supplied by PAASS-LC will decode this file and supply the data to my program. This raw data will be energy calibrated as discussed in section 2.2 above. This calibrated data will then be will then be

gated with respect to time to find the relevant coincident events. This coincident data will then be graphed using the ROOT programming language.

4.3 Interface

This be a web application where a user may submit their ldf file. They may also record a new ldf file using the poll2 tool. They will then process this recording using the coincidence processer written for this project. The resulting graphs will be displayed on the interface and root files will be available to download to their respective computers for further analysis.

4.4 Configuration

The UTKscan tool is configured using an xml file. The users will be allowed to submit their own configuration files. Or users will be allowed to use a default config file supplied by the interface.

5 REQUIREMENTS ANALYSIS

5.1 Functional Requirements

The user must be able to record radioactive decay events using the PAASS-LC framework which is designed to be function using the pixie16 data acquisition system. Data should be recorded using NaI scintillator detectors. Data will need to be analyzed, generating graphs which are both calibrated in energy and showing only events which coincide with one another.

5.2 Configurations

Configuration files will need to be able to set

- The current channels to monitor for data.
- The current slot the pixie16 card is installed in.
- The parameters regarding the coincidence data being analyzed.
- Time calibration parameters.
- Time window size
- Energy data to be used when generating coincidence spectrograms
- Calibration data should be supplied
- Calibration constants. (Calibration will not be redone)
- Data required for the Calibration in section 2.2 above

5.3 Data Output

Graphs generated should be displayed on the interface. Such as:

- Spectrums for each detector individually
- Spectrums for coincidence data

As well as the data used for generating these graphs as a ROOT file, which then may be downloaded to the experimenter's computer for further analysis.

6 PROJECT PLANS

Term 1:

- Documentation
- Gathering requirements
- Proposal

Term 2:

- Prototyping
- Delivery of proof of concept
- Working coincidence processor

Term 3:

- Web interface implementation
- Testing
- Feedback from stakeholders

Term 4:

- Implementation of feedback
- delivery of final code
- testing of code at IThemba labs for coincidence experiment

7 APPLCATIONS

7.1 Nuclear Medicine

Positron emission tomography (PET) is a powerful and non-invasive method of imaging physiological processes occurring in the body. The time of flight techniques spoken about in this paper are already being used in modern PET scanners as, "In the newer generation of PET detectors the resolution of the tomographic image is improved by determination of the annihilation point along the line-of-response [5]." This method is powerful as it reduces the noise along the line-of-response. This noise reduction occurs as the double event from the single particle decay reduces background noise as events recorded along the line-of-response which do not occur within a certain time frame are ignored, thus reducing the false positives caused by background radiation.

8 DESIGN

8.1 Data/Class Design

There are various data models used by both the web front end and only a single data model used by the coincidence processor

8.1.1 Coincidence Processor Data Model. The eventProc model is the most important, yet simplest, data object in the entire project. This object has these following fields

{
EnergyChannel : double,
TimeInClockCycles : double,
Slot: int,
Channel: int
}
Both EnergyChannel, and TimeInClockCycles fields need processing to be meaningful in the context of this processor code. The energy field needs to be calibrated using the method described in section 2.2. The time field needs to be calibrated using the frequency of the XIA data logger, which in our case is a 250MHz system. Which means that each clock cycle has a period of 4ns. The Slot and Channel field keep track of the specific detector being used as 2 detectors can be attached to different pixie-16 cards inserted into the same XIA cage.

8.1.2 Web Interface. Some metadata about an experiment is recorded by the webpage and is structured according to the metadata model.
{
ExperimenterName: String,
ExperimentDate: DateTime,
ExperimentName: String,
ExperimentShortDescription: String,
ExperimentLongDescription: String,

}

8.1.3 Coincidence Processor Class Description. The Coincidence processor consists of a single class for detecting coincidence data. The Class Diagram can be found below.

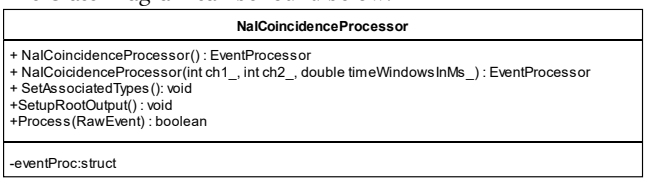

Figure 3: Class Diagram depicting the functions and variable of the NaICoincidenceProcessor class.

8.2 Web Front End Data Model

The web front end will have various data models, which describe the various states of the stages of the application. The first will be the model to manage the configuration, datafile pair.
{
ConfigurationFilePath : string,
DataFilePath : string,
DestinationFilePath : string,
State : IState,
}
The DataFilePath is the location of the data file, after being uploaded to the server, which is the raw data file that will be processed. The ConfigurationFilePath is the location of the configuration file, after being uploaded to the server, which is responsible for telling PAASS-LC which processors to be used. The DestinationFilePath will be used to tell PAASS-LC where to store the generated root files. The state maintains the state of the current run of the PAASS-LC program, the state field calls the IState object which looks like
{
IsRunning : boolean,
HasError : boolean,
OutputMesasge : string,
}
IsRunning tells the front end if the PAASS-LC program is still executing. HasError tell the front end if the PAASS-LC program has run into an Error and output message tells the front end the messages given by the PAASS-LC program.

8.3 Architectural Design

The Architectural design of this application is built to facilitate the flow described by the interface design. The project will be built in two parts.

8.3.1 The Coincidence Processor. The coincidence processor will be built into PAASS-LC as all other processor codes are for this framework. This allows me to leverage the power of this Package such as reading captured data, and easily plotting graphs.
As PAASS-LC sends each individual recorded event to our processor code we will need to have an external store of events to accumulate all events which are found.
After this point it will become possible to search for events which are in coincidence. An event in coincidence must be an event found at exactly 2 detectors and these 2 events must arrive at each detector within a specific time window of one another. The Time window as well as the detectors pairs being monitored should be set in the configuration file.

8.3.2 The Web Front End. The web front end will be used to facilitate the easy flow of data from capture to processing and finally to the retrieval of human readable data. The front end will do absolutely no processing.
The web front end will use a NodeJS, express, webserver to allow an experimenter to upload a configuration file as well as the raw experiment data. NodeJS also allows for a program to execute programs using Linux terminal commands. This is important as it will allow for the PAASS-LC framework to be called. It also allows for files to be read and be sent back to client devices.
I chose to have a Web front end as it allows many experiments to interact with DAQ stack using their own computers while also making it possible for physicist with a minimal understanding of the Linux environment to still interact with this tool.

8.4 Interface Design

The usage of the DAQ stack Coincidence processor will follow a logical flow. This will flow in the manner described in number form below. The below flow is related directly to using the coincidence processor.

1. The user will upload the configuration and data files.
2. PAASS-LC will call the corresponding processors specified the configuration file.

3. When the coincidence processor is called the events coming from the detector will be stored in the event store component.
4. once all the events from the current run has been stored in the event store the time coincidence processor will find all events which are in coincidence.
5. PAASS-LC convert the plots generated in the coincidence processor to a .root file which can then be downloaded from the processed file download component.
6. While the user is waiting the web, front end will display the loading component to indicate the file is currently being uploaded or the file is currently being processed

8.5 Component Level Design

The DAQ stack Tool will be built using the PAASS-LC framework to do coincidence detection. This will require the development of a small set of components to be developed. This will be both for the core coincidence detector code, as well as the web front end.

1. Coincidence processor
    a. Event store
    This contains information about each of the events given by the particle detector.

    b. Time coincidence processor
    This determines if 2 events have arrived at the different detectors in coincidence

2. Web front end
    a. Configuration, and data file upload
    This tells PAASS-LC what the data is and how to process it.

    b. Run Processor
    This component initiates PAASS-LC using the information provided in a.

    c. Processed file Download
    This allows the end user to download the processed data files in a .root file format

    d. Loading component
    This is used to block the user from initiating another execution of the PAASS-LC program while simultaneously informing the end user that the PAASS-LC program is currently processing their data.

9 PROTOTYPE

9.1 Processor Code

For this project a prototype has been developed to demonstrate the ability of the coincidence processor. This processor takes data from precisely 2 detectors and searches for 2 events which have been detected in coincidence. It then plots the energy spectrum of the coincidence events using the energy given by the first detector.

9.2 Hardware

The XIA system is setup to use the pixie16 data acquisition modules. These modules record data coming from 2 NaI Scintillator detectors. These detectors monitor the Gama rays emitted during the radioactive decay of Co60

9.3 Software

A spectrogram is generated for each detector which can be found in Figure 4.a and 4.b and a coincidence spectrogram is generated from events which occurred within a certain time window of each other. This can be found in Figure 4.c
A positive outcome for this experiment is that a spectrogram with the same shape as that of either detectors but also with having fewer events than either detectors would have respectively.
For this prototype I noticed an issue with the way that the PAASS-LC framework was interpreting data that was coming from our particle detector. This led me to investigate the issue and ultimately, I got in contact with the developers of this program and managed to fix the issue and this bug was fixed in the latest published version of PAASS-LC.

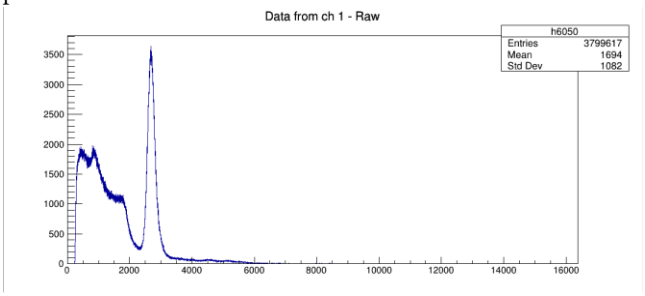

Figure 4.a: A energy spectrogram for the detector to the left of the radioactive source. (counts / channel)

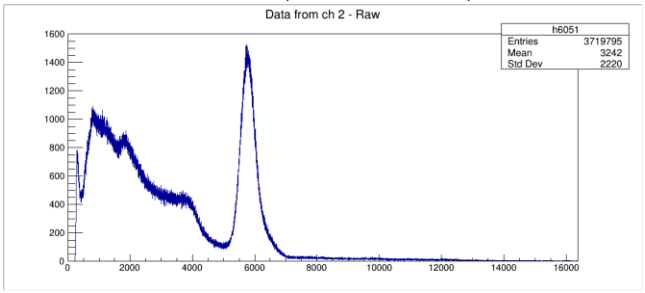

Figure 4.b: A energy spectrogram for the detector to the right of the radioactive source. (counts / channel)

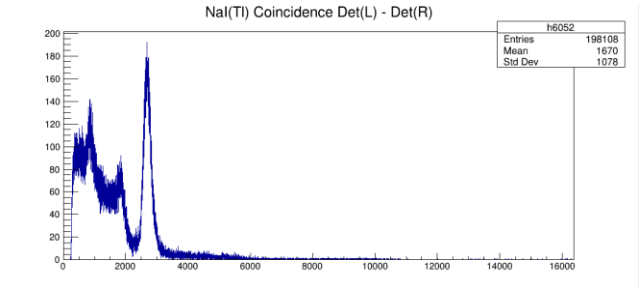

Figure 4.c: A energy spectrogram for the coincidence events happening between Figure 4.a and Figure 4.b. (counts / channel)

For my prototype an experiment was run in the physics building. The experiment resulted in the creation of Figures 4.a and 4.b. Our coincidence code was able to generate Figure 4.c using a timing window of 1ms and using the energy channel defined by the detector attached to ch1. Therefore, the pattern of the spectrogram follows that of Figure 4.a.

10 IMPLEMENTATION

10.1 Software And Hardware Requirements

10.1.1 Software for processor code. The software tools used for this project is extensive. The main framework used is PAASS-LC which relies on the root programming language developed at cern. Root is used as "It provides all the functionalities needed to deal with big data processing, statistical analysis, visualization and storage." [6] Root is built using make and is written in c/c++, therefor the gcc compiler is required. It uses the environment scripts to manage the installed system therefor these environment variables were sourced at boot. PAASS-LC is distributed through GitHub therefor the git version control tool is needed. PAASS-LC is built using CMake and compiled using make. PAASS-LC communicates with our pixie-16 system using a PLX controller therefor a PLX SDK is needed. The developer of PAASS-LC supplies the drivers for this system through his GitHub profile. The PLX controller service is managed using the popular Linux tool systemd. The XIA api tool is used to communicate with the pixie-16 system through the PLX controller and is used with the xia firmware.
For the
10.1.2 Software for web interface. The web interface is built using the react web framework. It is written using the typescript language. Material-UI is used to provided pre-styled web components such as buttons and text fields. Storage is provided by a json file. Both the frontend and backend are hosted using node server. Communication between the frontend and backend is managed using the axios library. Graphs are visualized using the plotly.js plotting library. File uploads are allowed using react-dropzone. The frontend is designed to be viewed using the Firefox web browser. Styling is accomplished using css. The backend api is hosted using the express web framework. Filesystem management is handled by the fs package and time is managed by the moment package. React router is used to manage the multiple webpages, as react is designed for single page web apps. All code written for this project is done using the Visual Studios Code IDE. All code is shared using git.
10.1.3 Hardware. The processing code, backend and frontend is hosted using a server located in the MANDELA lab. The server is connected to the pixie-16 system using a PLX controller. The pixie-16 system receives signals from 2 NaI scintillator gamma detector. A High Voltage power supply is used to bias the detectors. A pulsar is used for testing and produces idealized waveforms for signals coming from the NaI detector. An Oscilloscope is used to study the waveform of the signal coming from the NaI detectors to ascertain the properties required by PAASS-LC for signal processing.

10.2 Functions Methods And Classes

10.2.1 Processor Code. The Event model described in section 8.1.1 is implemented in the header file NaICoincidenceProcessor.hpp. All coincidence processing occurs in the NaICoicidenceProcessor.cpp file. As both files are experiment processor codes, they are stored in the experiment processor directories within PAASS-LC. The histograms for the start, stop detectors and the coincidence data is registered with root in the DeclarePlots method. The processor tells PAASS-LC that it is expecting data from a NaI scintillator detector in the SetAssociatedTypes method. All processing is done in the Process method. The event data is given by PAASS-LC using a vector. Therefor the events from the start and stop detectors is separated into 2 vectors of type eventProc. Then a loop is used to get elements from both vectors and the time difference is compared. During the comparison stage the data is written to the root histograms as well as a text file which is displayed in the web interface.
10.2.2 web interface. As the project uses the react framework it follows the standard react file structure. Images are stored in the asset's directory. The process, display and history webpages are stored in the module's directory. Components which are generic between pages are stored in the components directory and components which are specific to a page are stored in the same directory as its corresponding module. File upload for data files and the configuration panel for PAASS-LC are done in the process webpage. After the backend processes the data file. Data will be added to the history page. The history page lists all past experiments and a specific experiment with their metadata can be visualized in the display page. In the history page all experiments are displayed in Cards displaying ExperimenterName, ExperimentDate, and ExperimentShortDescription, as well as the plot for the histogram recorded for that experiment. The histograms are rendered by the component histogram. The visualize page is used to display all the data available on the history page but also shows the ExperimentLongDescription and allows the user to download the processed root file aswell as the raw data file used to generate it. The frontend and backend communicate using HTTP post and get requests. All communication from the frontend is facilitated by a services class called communication.ts located in the services directory.
10.2.3 backend. All endpoints for express is handled in the index.ts file. Each operation is managed by separate endpoints. The endpoint processExperiment is used to run the coincidence processor code. It executes PAASS-LC and stores the data in the

processed directory. All raw experiment data is stored in the experimentData directory. All metadata for the coincidence data is stored in the DB.json file. Interactions with the file system occur using the fileManagement class and interactions with the DB.json file occur using the storage class.

10.3 Limited Testing

10.3.1 Processor Code. To test CFD triggering we plotted the start vs stop times of our events which were in coincidence to measure how well the technique would benefit the project.

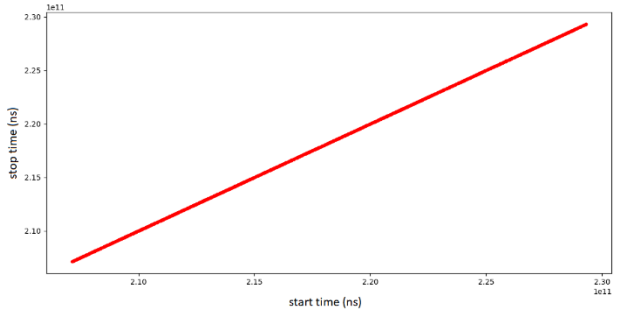

Figure 5.a: Start time vs stop time. Start time increasing to the left. Stop time increasing upward. Experiment was run for 3000 seconds. Using CFD triggering.

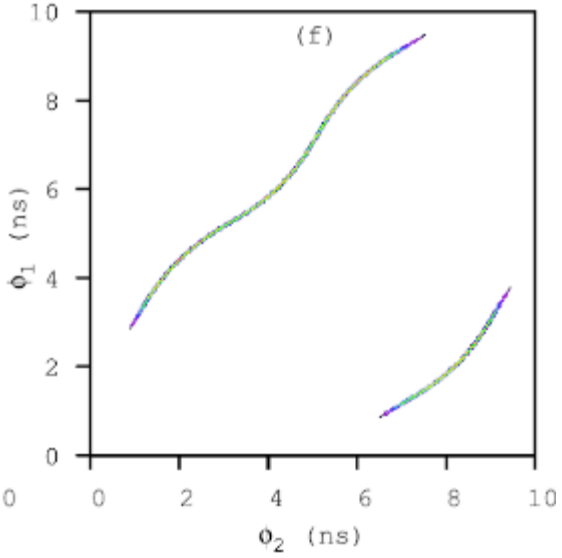

Figure 5.b: Ideal graph for the Start time vs stop time. Start time increasing to the left. Stop time increasing upward. Experiment was run for 10 nano seconds. Using CFD triggering. [7]

As the Figure 5.a is incredibly linear while Figure 5.b is less so proving that there is an issue using CFD triggering. This graph shows us that there is likely a feedback loop occurring in the system. Under closer inspection it was found that the signal traces recorded by PAASS-LC we far too short. The trace was not long enough to fall back to the ground state. Due to its long signal decay time of the NaI detector. As the entire trace length is needed for the CFD algorithm to model the trace, it was decided that we use simpler leading-edge triggering despite its lower time resolution.

A Coincidence experiment was run, and a spectrogram of the start time subtracted from the stop time spectrograph was generated.

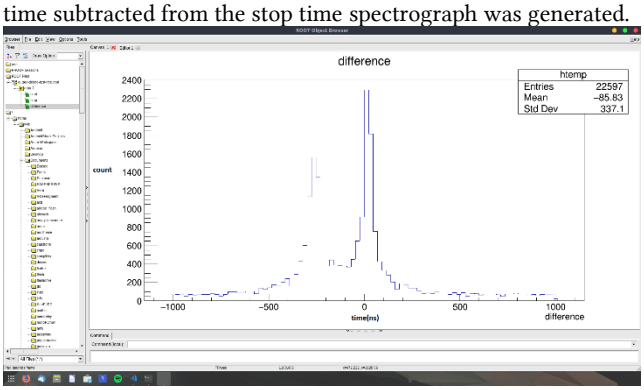

Figure 6.a: Start time subtracted from stop time. Time increasing to the right. Amount of start time subtracted from stop time increasing upwards.

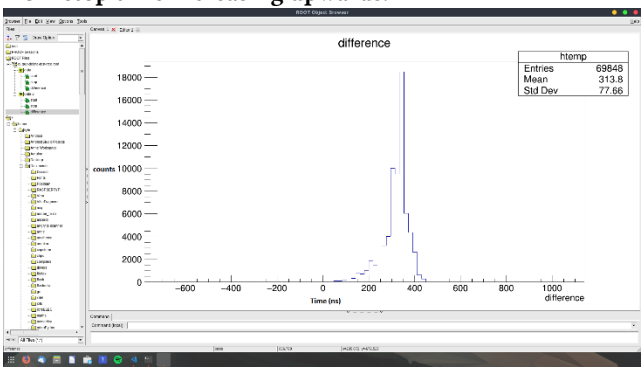

Figure 6.b: Start time subtracted from stop time. Time increasing to the right. Amount of start time subtracted from stop time increasing upwards.

While a gaussian distribution was expected figure 6.a clearly shows that there is an issue with the data being collected from the detectors. The trace length was subsequently lengthened, and the leading-edge trigger threshold was optimized during the acquisition phase. This new experiment data is then processed resulting in the gaussian distribution observed in Figure 6.b. “Because of the usually unknown processing times of the electronic components, it can hap-pen that the logic signals do not reach the coincidence unit simultaneously, even though the two gammas reached the detectors at the same time.” [8] As the gaussian is not centered at zero it shows that there is a signal delay between the start and stop detectors of approximately 320 ns. Therefor coincidence data can now be processed. Using this known time delay.

10.3.2 Web Interface. The web interface is far less complicated and only requires testing the file upload. This is done by uploading configuration files of incorrect file formats. The graphs generated by the processing code will always generate data in a standard format and therefor testing the plotting of data files will not be required. Also, the coincidence processor code is run both manually as well as using the web front end to verify consistency between the 2 methods.

11 CONCLUSIONS

The tool that we will be developing has numerous applications in the fields of not only particle physics but also Nuclear medicine. This tool will be developed throughout the year and will be tested, and its operation be verified in an experiment run at iThemba labs later this year. It will consist of the modification of an existing widely used scientific tool. It consists of interfacing with embedded systems, and FPGA's. Various modern Optimization Algorithms will be implemented and studied throughout this development cycle. The interface will use modern web frameworks to build a high performance and easy to use fully capable digital data acquisition system. The prototype created for the initial phases of this project provides promising results with respect to the viability using the hardware stack for coincidence experiments. Future experiments and optimization should improve this method greatly such as improved timing resolution of the detector and higher frequency data acquisition hardware.

References

[1] C. Fabjan, "Detectors for elementary particle physics," Cern, Geneva, Switzerland, 1994.

[2] S. Paulauskas, "PAASS-LC," 13 February 2018. [Online]. Available: https://github.com/spaulaus/paass-lc/releases. [Accessed 25 March 2019].

[3] E. Prince, "Well Defined Energy emmissions for various isotopes," in *International Tables for Crystallography*, United States, Wiley, 2004.

[4] P. Siegal, "Data Analysis, Calibration of Equipment," in *Radiation Biology Lecture Notes and Lab Experiments*, California, California State Polytechnic University, 2016.

[5] M. Silarski, "A novel method for calibration and monitoring of time synchroni-zation of TOF-PET scanners by means of cosmic rays," Institute of Physics, Jagiellonian University, Poland, 2013.

[6] CERN, "Root," Cern, 02 February 2019. [Online]. Available: https://root.cern.ch/. [Accessed 08 September 2019].

[7] S. V. Paulauskas, M. Madurga, R. Grzywacz, D. Miller, S. Padgett and H. Tan, "A digital data acquisition framework for the Versatile Array of NeutronDetectors at Low Energy (VANDLE)," *Nuclear Instruments and Methods inPhysics Research Section A: Accelerators, Spectrometers, Detectors and associated Equipment*, p. 25, 2013.

[8] K. Panda and T. Gregor, "Measuring the time spectrum," Schuler Labor, 2013.