CERN ProtoDUNE Single Phase Cosmic Ray Tagger DAQ

L. Elder[1]

[1]Roanoke College, Salem, Virginia

July 2020 - Prepared as part of the VT-Neutrino-REU-2020

Abstract

During this summer REU program, I have worked in preparing and commissioning a new C++ DAQ for the ProtoDUNE Single Phase Cosmic Ray Tagger (CRT). The CRT is installed in front and back of the ProtoDUNE Single Phase Time Projection Chamber (TPC) in the CERN neutrino platform and it is used to analyze beam halo and to identify a pure set of muons to be used for calibration purposes. I began by examining all parts of the new DAQ code and identifying how all subroutines are interconnected. Each function within the program was analyzed to understand its purpose and how it works. We have tested the new DAQ program using a setup built in the Physics Department at Virginia Tech, in Robeson Hall Room 1. The setup consists of two USB boards and a Linux computer. One of the USB boards is connected to a single multi anode photomultiplier readout board (MAPMT) and the other is connected to seven MAPMT boards. The system was run in various configurations to verify the various functionality of the DAQ. Initially, the code was run with only the USB connected to a single board. After verifying that everything was working properly with this setup, the other USB board with the seven boards was connected as well and the two USB streams were run simultaneously. The data collected was analyzed in detail to identify any potential problems. Once all the problems were eliminated, the code was tested with different data acquisition rates using a self-triggering option. At the end, we were able to successfully run the code with two USB streams and eight total MAPMT boards.

1 Introduction

Neutrinos are unique particles deserving of significant scientific study because they are the first evidence of physics beyond the standard model. They are extremely light with a mass orders of magnitude smaller than any other fermions [1]. As a result of their small mass, no exact measurement of their masses can be determined. As their name suggests, they also have no charge and interact extremely weakly with other particles. For a neutrino with energy E_ν 1 MeV traveling on a trajectory through the Earth's center, the probability of an interaction is about 10^{-11} [1]. They were first theorized by Pauli in 1930 but due to their small size and weak interactions, they were not detected until 1956 when Reines and Cowan successfully measured them [2]. Despite their only relatively recent discovery, they are extremely plentiful in the universe around us. Every second, around 60 billion neutrinos produced by the sun, travel through each square centimeter of someone's body [1].

All neutrinos have left-handed spin, and all anti-neutrinos have right-handed spin [1]. This was a significant discovery because it violated two symmetries which were believed to be absolute: the charge (C) symmetry and the parity (P) symmetry [3]. This apparent contradiction was resolved with the idea of CP-symmetry which combined C and P symmetries. However, violation of CP-symmetry has since been observed in other particles. CP-violation has not yet been observed in neutrinos. CP-violation clearly differentiates between matter and antimatter. As a result, it is possible that CP-violation could be the cause of the imbalance of matter and antimatter in the universe. Learning more about CP-violation and observing it in neutrinos could lead to a better understanding the great mystery in physics of why matter dominated over antimatter.

There are three different flavors of neutrinos each corresponding to the different type of leptons: ν_e, ν_μ, and ν_τ corresponding respectively to electron, muon and tau. Eventually, due to the results of several

experiments, it was determined that neutrinos oscillate between the different flavors. However, this is only possible if neutrinos have mass, contradicting the Standard Model which stated that they had no mass [4]. In addition to the three different neutrino flavors, there are three neutrino mass eigenstates. A neutrino of a specific mass could be any one of the neutrino flavors and likewise, a neutrino of a specific flavor does not have a specific mass [4]. The different flavors are formed by the combination of the mass states and in the quark sector. This mechanism allows neutrinos travelling through space to oscillate between the different flavors.

The Deep Underground Neutrino Experiment (DUNE) is a massive neutrino experiment designed to observe neutrino oscillation and try to measure CP violation in the neutrino sector. It is believed that a more precise measurement of neutrino oscillation could lead to significant scientific discovery. One major goal is to observe CP-violation to hopefully gain an understanding of why the universe is matter-dominated. DUNE will also measure neutrinos from core-collapse supernovae to gain insight into the nuclear process that cause and govern those explosions [5]. In DUNE, an intense neutrino beam will be sent from the Fermi National Accelerator Laboratory in Illinois, to a suite of detectors placed deep underground about 1,300 km away in the Sanford Underground Research Laboratory in South Dakota [5]. The DUNE experiment will also be equipped with a near Detector, located at Fermilab around 500 meters away from the neutrino source [6]. The Far Detector (FD), located at Sanford Lab, will consist of four Liquid Argon Time Projection Chambers (LArTPC) containing approximately 68 kilotons of LAr in total [5].

A substantial prototype program (ProtoDUNE) has been established at CERN to design and test the two different types of LArTPC detector: Single-Phase (SP) and Dual-Phase (DP) [5]. In DP detectors ionization charges drift through the LAr and then a gas phase where they are amplified while SP detectors do not contain a gas phase [7]. Located at the CERN Neutrino Platform facility, the ProtoDUNE prototypes are used to test the designs for the DUNE detector components, and to determine how to install and run those detectors [5]. Additionally, the response of the detectors to different particles at varying energies will also be measured [5]. The ProtoDUNE-SP detector, which began collecting data at the end of 2018, is extremely important in the development of the Single-Phase DUNE FD module. It is currently the largest monolithic SP LArTPC detector in the world with around 0.77 kton of LAr [8]. Detectors of this scale present large challenges making ProtoDUNE-SP essential to the success of DUNE.

2 ProtoDUNE-SP detector

2.1 Goals

The overall goal ProtoDUNE-SP is to help with the development of the DUNE far detector. In order to be fully successful, there are four main objectives that ProtoDUNE-SP seeks to complete:

- Develop good procedures for producing and installing the single-phase far detector.
- Use cosmic-ray data to ensure the detector design performs at an adequate level
- Collect large amounts of test-beam data in order to understand and properly calibrate the detectors response to a variety of particles.
- Ensure the long-term stability of the detector with the purpose of limiting the risks associated with building the much larger DUNE FD [8]. By accomplishing these goals, ProtoDUNE-SP can aid DUNE significantly and push the scientific community closer to potentially significant scientific discovery.

2.2 Detector

Pictured in Figure 1 is the ProtoDUNE SP TPC, the detector consists of two drift volumes on either side of a central cathode plane and inside two anode planes [8]. This setup, which has an overall active volume of 7.2 m deep, 6 m high and 7 m wide, is surrounded by a field cage (FC) [8]. Each anode plane consists of three Anode Plane Assemblies each 6 m high and 2.3 m wide [8]. The APAs each contain three planes of sense and shielding wires which are supported by a frame [8].

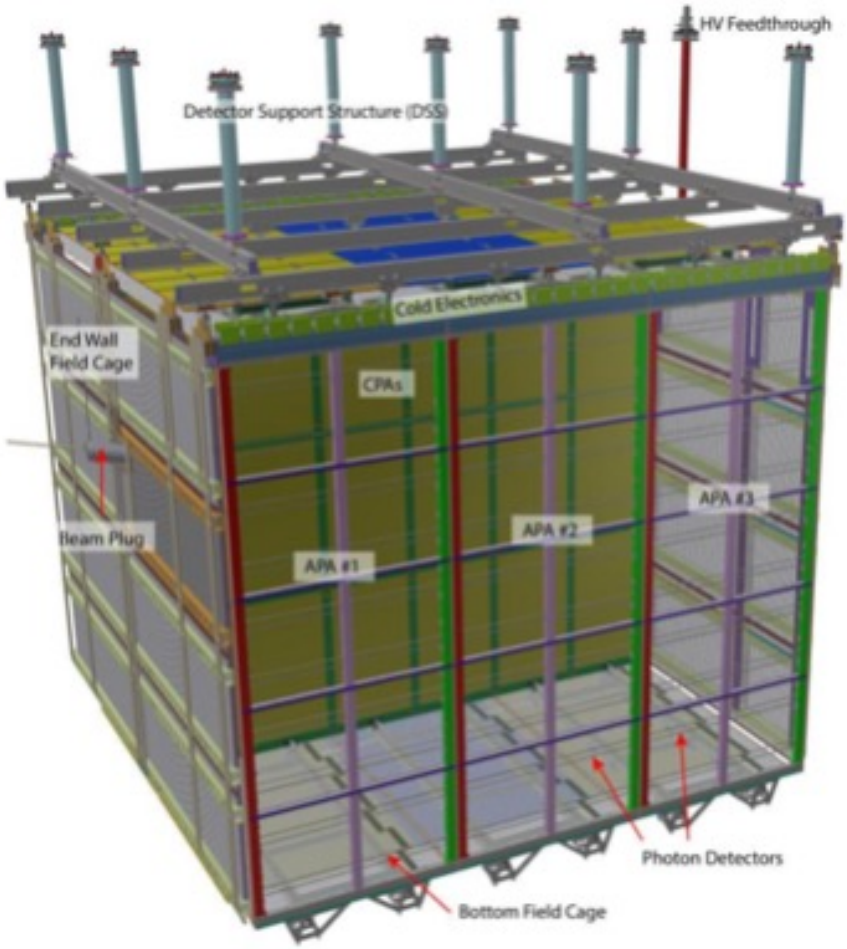

Figure 1: Schematics of the ProtoDUNE-SP TPC with in evidence of the anode plane assemble (APA) numbering scheme, beam plug and high voltage feed-through. Picture taken from [8].

The cathode plane is made up of the Cathode Plane Assemble (CPA) which contains 18 CPA modules [8]. The CPA modules, which are each 1.18 m wide and 2 m high and are arranged six wide and three high [8]. The CPA produces the 500 V/cm drift field while the FC ensures that the electric field remains uniform throughout the drift regions [8].

The Cold Electronics (CE) digitizes and amplifies the signals on the sense wires and then sends them to the Data Acquisition system (DAQ) [8]. The CE are mounted on the APA frame allowing them to be immersed in the LAr [8]. The data from the DAQ are transmitted to the central CERN Tier-0 Computing Center and to other partner sites to be processed and analyzed [8].

3 Cosmic Ray Tagger

3.1 Detector

The Cosmic Ray Tagger, which is installed vertically in front and back of the ProtoDUNE-SP TPC, plays a crucial role. The CRT is used to tag muons travelling through the TPC and can detect muons at shallow angles. The tagged muons consist of cosmic ray muons and beam halo muons which are used to analyze and reduce the impact of space charge effects [8]. In addition, the CRT is used for calibration, analyzing beam halo and to help measure the lifetime of electrons in the TPC [8].

The CRT consists of several scintillator-strip modules with the electronics necessary to collect data. Each

module is made up of 64 scintillator strips, each 5 cm wide by 1 cm thick by 320 cm long [8]. The strips are arranged into two parallel 32-strip layers offset by one half strip width [8]. A wavelength-shifting fiber with a 1.5-mm diameter is inserted in the center scintillator strip [8]. The resulting modules, which are covered with aluminum, have a size of approximately 360 cm long by 162.5 cm wide and they are 2 cm thick [8].

One end of the 64 wavelength-shifting fibers is connected to a Hamamatsu M64 multi-anode photomultiplier tube (PMT) while the other end of each fiber is mirrored to increase light collection at the PMT end [8]. A custom front-end board containing a MAROC2 ASIC and an FPGA is connected to each M64 [8]. A multiplexed 12-bit ADC is used to process signals that surpass a common threshold and give from this give pulse height information for hit strips [8].

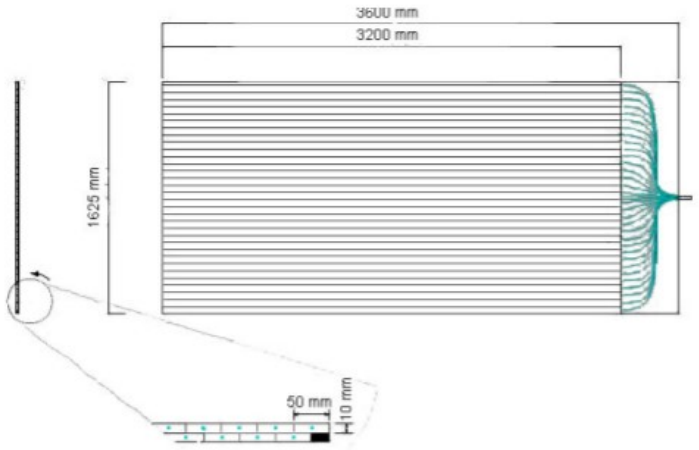

Figure 2: Illustration of a 64 scintillator strip CRT module as described in the paragraphs above. Picture taken from [8].

The ProtoDUNE-SP CRT consists of units of four modules arranged into two orthogonal layers each made up of two modules side by side. This provides full geometrical information in the x and y. The resulting area of each unit is 320 cm by 320 cm [8]. Four of these units are used on both the upstream and downstream sides of the TPC. On the downstream side the four units are arranged in a single panel and set up directly behind the TPC. On the upstream side the units are arranged in two panel of two units each. The left panel is set up directly in front of the TPC while the right panel is located 10 m back due to the location of the neutrino beam pipe and its supporting structure.

3.2 CRT DAQ

A series of ADC values for hits above a threshold value together with time stamps are produced by the CRT readout. The time stamps in conjunction with the ProtoDUNE-SP sync pulse and global clock (50 MHz) are used to merge the CRT and TPC information [8]. The CRT DAQ software framework is based in the C++ programming language and consists of several files within a folder called cpp_readout.

The oneboard.cpp and sevenboard.cpp are the two used to collect data from the PMT boards. The CRT.cpp program contains most of the functions used in the oneboard and sevenboard programs. Many of the functions in the CRT.cpp program use functions contained in the usbreadout.cpp program to give commands to the USB and PMT boards. The backup folder is used to store original copies of the files so that if a file is changed with a negative outcome, there is something to go back to. Makefile is used to easily compile the CRT.cpp and usbreadout.cpp programs. The other files have various roles within functions in CRT.cpp.

The sevenboard program can be used to run using the USB stream connected to seven PMT boards. The oneboard program can be run with either USB stream individually, or with both simultaneously. Each program has the same overall structure, but due to its additional functionality, the oneboard program was used for all testing. The following shows how the oneboard program works, going through each major function individually.

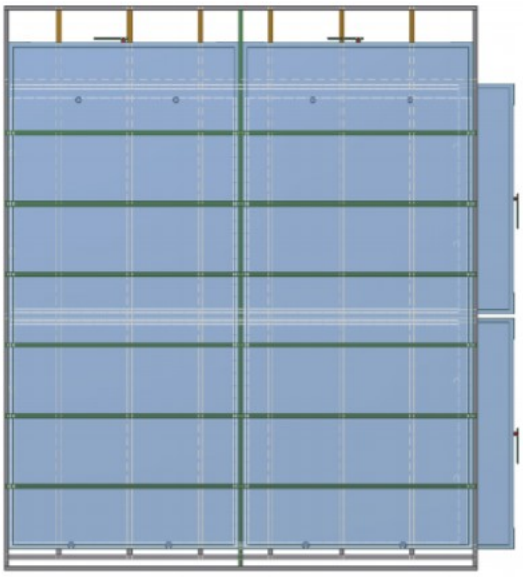

Figure 3: Drawing of one four-module CRT unit. The two units in the front are oriented vertically with the PMTs on top while the two units in the back are oriented horizontally with the PMTs on the right. The unit is held together by a support structure in grey and green in the picture. Picture taken from [8].

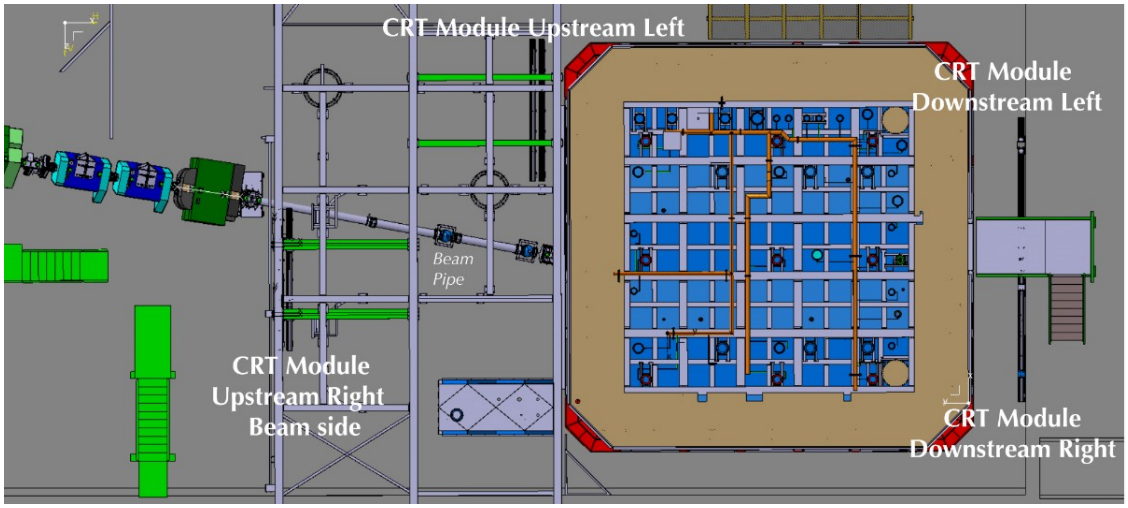

Figure 4: Layout of the CRT modules around the ProtoDUNE-SP TPC. The downstream side contains a single panel of four CRT modules located at the far right of the picture. The two modules on the left side of the upstream side are placed close to the TPC. However the beam pipe is in the way of the right modules forcing them to be located 10 m further away from the TPC.

Initializing Settings

The oneboard program uses the following function to initialize the settings for each PMT board:

loadpmtdata_auto(int usb, int pmt, int pmtserialnumber, string mysql_table)

The function is run separately for each PMT board. The first three inputs specify the USB number, the PMT board number and the PMT serial number, respectively. There are two modes available for loading settings: debug and mysql. If pmtserialnumber has no defined value debug mode is used. When debug mode is used, default settings are loaded directly from the function. Otherwise, if pmtserialnumber, mysql mode is used. The system will connect to the MySQL table specified by the final input and load settings from

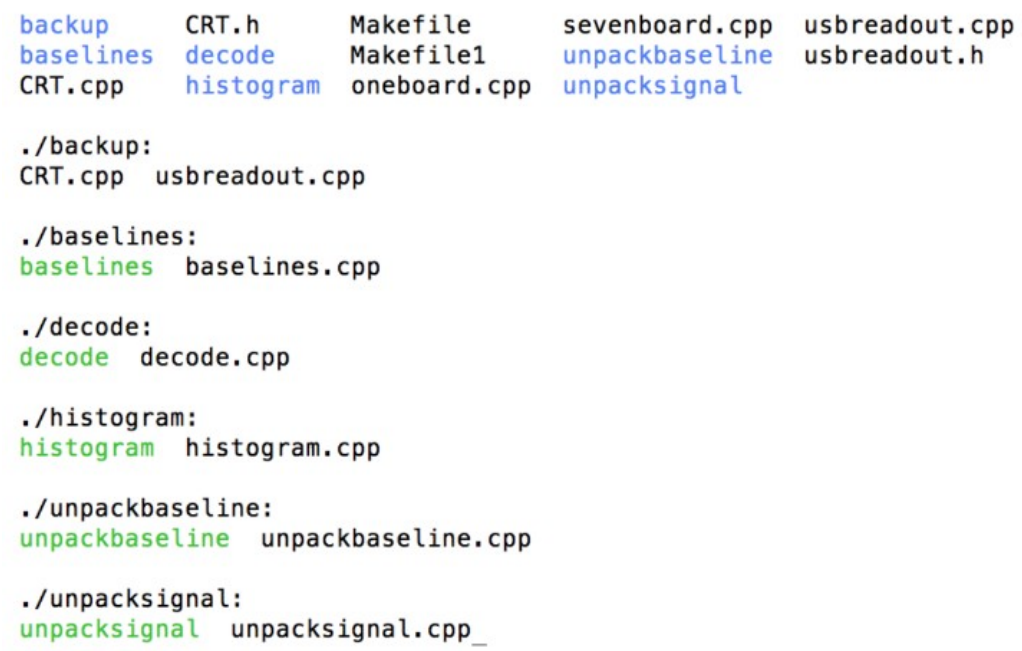

Figure 5: A layout of all the files used in the CRT DAQ C++ code.

that table. mysql mode allows for the settings to be easily changed by updated the MySQL table without having to change and recompile the code.

Taking baseline data and initializing the PMT boards

The following function is used to accomplish these tasks:

baselinemb(int disk_num)

The function begins by creating a data folder for the run and setting the run number. The input disk_num helps specify where the data folder will be located. Then baseline data, which contains information about the level of noise in each channel, is collected individually for each PMT board. baselinemb then calls the following function to initialize each PMT board for data taking:

initializeusb(int usb, int pmt)

This function is run for each PMT board with the inputs specifying the USB number and PMT number for each PMT board.

Taking data

The oneboard program uses the following function to collect data:

takedatamb(int runlenght)

The function begins by enabling the trigger for each PMT board to start taking data. The inhibit for writing data is released so the data can be recorded. The program then waits for a length of time specified by the input runlenght. All testing was completed using a run length of ten seconds. After waiting the appropriate time, the trigger for each PMT board is disabled and data writing is inhibited to complete the data taking process.

Processing Data

The program uses the function below to process the data:

generatecsv2(int usb, int pmt)

As with several of the functions above, it is run for each PMT board with the inputs specifying the USB and PMT numbers. The function creates a baselines.dat file for the run and then calls the following function:

signal(string dir, string file, string baselines, int usb, int pmt_board, string homedir)

As with other functions above, usb and pmt_board are used to specify the USB and PMT numbers for each board. This function processes all the data collected and generates all needed plots. All generated plots are saved into folders specified by the inputs.

4 Results

A setup in Room 1 of Robeson Hall, the home of the Virginia Tech Physics Department, was used to test the CRT DAQ code. This setup used two USB streams, one (USB 33) connected to one MAPMT board and one (USB 17) to seven MAPMT boards (numbered 0 to 6). I could analyze the data collected using this setup to determine if the code was working properly. To access and run the code, I remotely connected to a Virginia Tech physics computer running Linux. From this computer, I could access and modify all parts of the code, run the code and access all data collected from runs.

The first test completed involved running the code with only USB 33 and the one PMT board connected to it. After looking at the readout and data for this test everything appeared to be working well. This was expected because the code had previously been working will with just the single board. I did not thoroughly analyze the data for this run because more useful analysis could be conducted later by comparing the data between multiple PMT boards.

When we tried to test the program with both USB streams, we found that USB 17 was unable to initialize the system and as a result, none of the PMT boards on it gave any data. However, this issue was resolved by power cycling the system. With both USB streams working I could now run the code with all eight PMT boards connected. I could then look through all baseline data and data collected to determine if the code was fully working.

After running the code, I began by analyzing the baseline data. Baseline data, which was collected for each of the 64 channels on each PMT board, contained the number of hits recorded per channel, the average pulse height for hits and the standard deviation of the pulse height. If there were no problems with the baseline data, the number of hits and their pulse heights would be similar across all channels on a PMT board and each board would have similar values. After analyzing the data. I found that the hits per channel were constant across all channels of each board and the values were all between 500 and 538. As a result, there appeared to be no problems with the baseline hits. The average pulse height and standard deviation needed to be relatively consistent with no outliers across all total channels. This was also verified with pulse height values all between 1499 ADC and 1593 ADC and standard deviation values all between 1.45 and 1.9 ADC values. As a result, I concluded that there were no problems with the baseline data.

USB number	PMT number	Lowest Average Pulse Height	Highest Average Pulse Height	Lowest Standard Deviation	Highest Standard Deviation	Hits per Channel
33	3	1527	1568	1.45	1.85	500
17	0	1501	1539	1.49	1.90	500
17	1	1511	1552	1.49	1.86	508
17	2	1523	1564	1.48	1.82	513
17	3	1499	1543	1.54	1.87	520
17	4	1505	1546	1.53	1.85	525
17	5	1510	1533	1.50	1.88	531
17	6	1549	1593	1.53	1.86	538

Table 1: Summary of baseline data. The first two columns specify the USB and PMT board numbers. The next four columns give the ranges of the average pulse height and standard deviation across all PMT boards. The final column gives the hits per channel for each PMT board.

The main data collected for the run was analyzed next by using the plots created by the signal function. We collected data using a particular trigger configuration in which the board self-trigger themselves with a fixed and known trigger rate. The trigger can be modified using the DAQ interface. The hits per channel were constant among all channels on a given PMT board. The board on USB 33 had 691 hits per channels. On USB 17, boards 1 through 6 had 722 hits per channel while PMT 0 had 5054 hits per channel, exactly 7 times the amount on each of the other boards. In addition, the average pulse height per channel on PMT 0 was roughly half that of the other channels. This clearly showed something was wrong with the plots leading us to believe there was likely some flaw in the code that created the plots. After tracing through the code,

we eventually found the error. In the unpacksignal.cpp program there was an if statement depending on the PMT number which became false when the PMt number was 0. By adding an additional statement to account for the possibility of the PMT number being 0, the problem appeared to be resolved. I then ran the program again and analyzed all data collected to verify that the problem was in fact solved.

USB number	PMT number	Hits Per Channel Before	Average Pulse Height Before	Hits Per Channel After	Average Pulse Height After
33	3	691	977.22	694	971.92
17	0	5054	473.24	717	860.67
17	1	722	928.66	717	920.60
17	2	722	983.72	717	983.54
17	3	722	861.31	717	879.43
17	4	722	958.33	717	931.04
17	5	722	854.79	717	833.02
17	6	722	955.74	717	940.49

Table 2: Summary of the data collected for a run before and after the problem with the code was resolved. The first two columns specify the USB and PMT board numbers. The next two columns give the hits per channel and average pulse height for the PMT boards before the issue was resolved. The numbers for USB 17 PMT 0 are clearly outliers. The final two columns give the data for after the problem was fixed.

After ensuring that the code for the CRT DAQ was working properly, we decided to test the code with different trigger rates. By changing the self-triggering rate parameter in the initializeusb DAQ function (force_trigger), I adjusted the rate and for each of the three setting ran the code so I could compare the results. The three settings correspond to a self-triggering rate of 1 msec, 16 msec or 256 msec. All previous tests had been conducted using the 16 msec setting. The initializeusb function is used after baseline data has been taken, so the baseline data was unaffected by this change. As expected, the pulse heights also remained unaltered. The run with the 16 msec setting had roughly the same number of hits per channel on each PMT board as previous tests had. When compared to the 16 msec run, the 1 msec run had roughly 16 times more hits per channel on average while the 256 msec run had roughly 16 time fewer hits per channel on average. This was the desired result, verifying that the code worked equally well with different data acquisition rates.

Force Readout	Msec	33-3 Hits	17-0 Hits	17-1 Hits	17-2 Hits	17-3 Hits	17-4 Hit	17-5 Hits	17-6 Hits
0b01	1	11040	11457	11457	11457	11457	11457	11457	11457
0b10	16	694	717	717	717	717	717	717	717
0b11	256	43	45	45	45	45	45	45	45

Table 3: Summary of data collected using different data acquisition rates. The first two columns give the force readout value and the time associated with each value. The remaining eight columns give the hits per channel for the PMT board specified in the column header.

After the successful conclusion of this project, the new DAQ code will be imported in the ProtoDUNE DAQ framework at CERN. The functionality of the new software framework will then be tested with the four USB streams and 32 MAPMTs used in the CRT of the ProtoDUNE-SP detector.

5 Conclusion

During this project, we tested and debugged the new C++ DAQ for the ProtoDUNE-SP CRT. After thoroughly looking through the code to understand how all parts of it were connected, I began testing the system

using a setup at Virginia Tech. I analyzed all data collected by the runs to determine if the system was working properly. When an issue with the data for PMT 0 was observed, we traced the problem through the code until we found and fixed its source. After determining that the code was now working as intended, I tested it with different trigger rates and compared the results. After extensively analyzing the data, I found that even with different trigger rates, the code was performing as expected. The success of this project will go toward improving the ProtoDUNE-SP CRT DAQ and thus have a positive impact on the ProtoDUNE and DUNE experiments. The DUNE experiment is the future of neutrino physics and hopes to answer some great mysteries of physics.

6 Acknowledgements

I would like to thank the Virginia Tech Center for Neutrino Physics for hosting this project. My success would not have been possible without the support of Dr. Camillo Mariani and Linjie Gu. Thank you to Betty Wilkins for her work to organize the REU. The work of Luke Elder was supported by the National Science Foundation REU grant number 479901.

References

[1] P. Lipari. Introduction to neutrino physics. In *1st CERN-CLAF School of High-Energy Physics*, pages 115–199, 5 2001.

[2] Guido Fantini, Andrea Gallo Rosso, Francesco Vissani, and Vanessa Zema. The formalism of neutrino oscillations: an introduction, 2018.

[3] Motivation: CP Violation. URL: https://www.nevis.columbia.edu/daedalus/motiv/cp.html.

[4] David Kaleko. Outer veto calibration for double chooz. 2010.

[5] D. Adams et al. The protodune-sp lartpc electronics production, commissioning, and performance, 2020.

[6] Mehedi Masud, Animesh Chatterjee, and Poonam Mehta. Probing cp violation signal at dune in presence of non-standard neutrino interactions, 2015.

[7] Flor de María Blaszczyk. Prototyping the world's largest liquid Argon TPC: ProtoDUNE Single Phase. URL: https://tu-dresden.de/mn/physik/iktp/ressourcen/dateien/seminar/2018_11_15-Flor_Blaszczyk.pdf?lang=en, 11 2018.

[8] B. Abi et al. The single-phase protodune technical design report, 2017.