

Saint Francis Xavier University

Detection of Cosmic Ray Muons with Teachspin and Cosmic Watch Muon Detectors



Maxwell J. MacNeil

Supervised by
Dr. Hossain Ahmed

In partial fulfillment for the degree of
Bachelor of Science in Physics (Advanced Major)

in the
Faculty of Science
Department of Physics

March 17, 2025

Declaration of Authorship

I, Maxwell MacNeil, am declaring that this paper and the work contained in it are my own. Further, I confirm that:

- This work was wholly done while in candidature for an advanced major degree at this University.
- Where any part of this paper has been submitted at this University or any other institution, this has been clearly stated.
- Where I have consulted published work from others, this is always clearly attributed
- Where I have quoted from the work of others, the source is always given. With the exception of these quotations, this paper is entirely my own work.
- I have acknowledged all main sources of help.
- Where this paper is based on work done jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signature: Maxwell MacNeil

Date: March 15th, 2025

Abstract

Muons, fundamental particles within the Standard Model of Particle Physics, are produced when cosmic rays interact with the Earth's upper atmosphere. These particles are constantly striking the Earth's surface and can be detected using scintillator-based detectors. In this project, two different detectors were used to perform muon flux experiments. The first, the Teachspin Muon Detector, was calibrated based on the well-established value for muon lifetime. The second detector, the Cosmic Watch Muon Detector, was specifically constructed for this project. Through this investigation, we hope to gain a greater understanding of how these detectors function and to set up experiments to further explore muon behavior.

Acknowledgments

I would thank the entire physics and engineering departments at StFX University for the incredible education I have received these last four years, they have been essential in helping me achieve my goals.

I would also like to thank my supervisor, Dr. Hossain Ahmed, whose enthusiasm and knowledge sparked my interest in particle physics. I truly appreciate his constant encouragement and support. The skills I have gained throughout this research leave me feeling well-prepared for my future endeavors.

To my classmates and friends in physics, for always being supportive and making my time at StFX so enjoyable.

I would like to acknowledge Matt Penner, whose previous work and continued support made this project possible. Additionally, I would like to thank Audrey Yang, whose collaboration on experiments throughout the project was extremely valuable

A special thank you to my parents, Graham and Liz, and my sister, Nicole, for encouraging me to pursue my passion for physics.

Finally, thank you to NSERC and Belle II for funding this project.

Contents

1	Introduction	8
2	Muon Physics and the Standard Model	10
2.1	History of the Standard Model	10
2.2	Cosmic Rays	13
2.3	Special Relativity	15
2.4	Muon Interactions with Matter	16
3	Teachspin and Cosmic Watch Muon Detectors	18
3.1	Teachspin Muon Detector	18
3.2	The Cosmic Watch Muon Detector	21
3.2.1	Cosmic Watch Scintillator	21
3.2.2	The Silicon Photomultiplier	21
3.2.3	Detector Electronics	23
4	Data Collection and Analysis	25
4.1	Teachspin Muon Detector	25
4.1.1	Example Data Interpretation	25
4.1.2	Calibration of Teachspin Muon Detector	26
4.1.3	Development of New Teachspin Muon Detector Software	27
4.1.4	Inconsistencies with Battery Trials	28
4.2	Cosmic Watch Muon Detector	30
4.2.1	Example Data Interpretation	30
4.2.2	Flux Measurements and Error Bars	31
4.2.3	Coincidence Measurements	32
4.2.4	Keppoch Experiment	32

5	Conclusion and Future Work	34
A	Python Code for Teachspin Detector	38
B	Python Code for Cosmic Watch Detector	43

List of Figures

2.1	The Standard Model [1].	12
2.2	Modified diagram showing that muon decay Feynman diagram. Adapted from [2].	13
2.3	An illustration showing the various decay chains as a result of cosmic rays hitting Earth's atmosphere [3].	14
3.1	Base of the Teachspin muon detector.	18
3.2	Diagram of a PMT [4].	19
3.3	Block Diagram for Teachspin Detector Circuit [5]	20
3.4	Cosmic Watch muon detector.	21
3.5	Scintillator partially wrapped in reflective foil.	22
3.6	Magnified view of the SiPM [3].	22
3.7	Cosmic Watch muon detector main PCB	23
3.8	Circuit diagram of Cosmic Watch muon detector [6].	24
4.1	Raw data from Teachspin detector	25
4.2	Decay histogram for measuring muon lifetime.	27
4.3	Mean muon flux plot	28
4.4	Current muon flux (updated every 10 minutes).	29
4.5	Inconsistent flux from battery trial.	29
4.6	Cosmic Watch muon detector example data.	30
4.7	Difference in mean muon flux between 3rd and 4th floor of Mulroney Hall.	31
4.8	Difference in muon flux between Keppoch mountain (220m) and reference point (70m)	33

Chapter 1

Introduction

Nearly everything is really interesting
if you go into it deeply enough.

Richard P. Feynman

Muons are fundamental particles within the Standard Model of particle physics. They are similar to electrons; except they are over 200 times larger and only exist for approximately 2.2 microseconds before decaying into an electron and a couple neutrinos [7]. What makes them particularly intriguing is their dual nature: as unstable particles that are created in high-energy environments, they provide valuable insight into fundamental physics, while simultaneously being abundant and relatively easy to detect [3]. This makes them ideal for undergraduate study, as students can design their own experiments without needing access to multi-billion-dollar particle accelerators.

I began this project with the goal of calibrating the StFX Physics Department's Teach-spin muon detector, to aid with experiments investigating muon flux at different altitudes. As my research I developed, I realized that building a second muon detector designed more specifically to muon flux experiments would be both feasible and beneficial. This new detector was built following the Cosmic Watch project [6], a guide developed by Dr. Spencer N. Axani and his colleagues at MIT, which instructs undergraduate students how to build their own muon detectors using mostly standard electronics components. Over the course of this project, I studied both detectors extensively and performed flux

experiments with both.

In this Advanced Major Project, I will provide a detailed overview on muons in the context of the Standard Model, offering the necessary theoretical background for understanding muon flux experiments. I will then outline the detection process for both detectors, comparing and contrasting their designs. A particular focus will be placed on the calibration of the Teachspin detector and the construction of the Cosmic Watch detector. Finally, I will discuss the experiments conducted as part of this project, present their results, and propose potential future work for these detectors.

Chapter 2

Muon Physics and the Standard Model

This section gives an overview of the history of the standard model based on Dr. David J. Griffith's [8] explanation, focusing on where muons fit within the context of the standard model. The discussion then explores the theoretical background needed for the rest of the paper.

2.1 History of the Standard Model

For millennia, humans have posed the question “What is everything made of?”. While philosophers like Democritus considered indivisible atoms [9], major scientific progress in answering this question began with J.J Thomson’s discovery of the electron in 1897. Thomson determined that cathode rays (a beam of electrons) emitted by a hot filament carried an electric charge. By measuring how the rays interacted with electric and magnetic fields, Thomson was able to determine the velocity and charge-to-mass ratio of electrons. The enormous charge-to-mass ratio suggested that either the charge of the particle was much larger, or the mass was much smaller than any known ion. Thomson correctly deduced that electrons were part of the structure of an atom; however, Thomson incorrectly thought the electrons were suspended in a positively charged “paste”, like plum pudding [10]. While his “plum pudding” model of the atom was incorrect, Thomson did give humanity its first look into the subatomic world. This event marked the birth of elementary particle physics.

A series of discoveries about subatomic particles and their interactions throughout much of the 20th century greatly enhanced the understanding of the natural world. Ernest Rutherford's scattering experiment demonstrated that most of the mass and the positive charge of an atom are concentrated in the core of the atom. This tiny core would come to be known as the nucleus, and this "nuclear model" of the atom would quickly replace Thomson's model. Rutherford coined the term "proton" for the nucleus of the hydrogen atom, which is the second subatomic particle discovered after the electron. This theory was further validated by experimental verification on Niel Bohr's model of hydrogen, which proposed that a hydrogen atom was a single electron orbiting a proton. Unfortunately, protons, and electrons could not explain the mass of heavier elements like helium and lithium, suggesting another missing piece in the atomic puzzle. James Chadwick resolved this discrepancy in 1932 with the discovery of the neutron. With the electron, proton, and the neutron, the internal structure of atoms had been revealed, thus concluding the classical period of particle physics.

Now that the components of the atom had been determined, a new question arose: how does the nucleus remain intact? In theory, the protons should repel because of the electromagnetic force. The proposed explanation was the strong force: a powerful force that acts at a very close range to hold the nucleus together. In 1934, Hideki Yukawa proposed a theory that stated this strong force would have to be mediated by a particle with a mass nearly 300 times larger than an electron, and approximately a one-sixth of the protons mass. This theoretical particle was classified as a meson, meaning middleweight. Similarly, the electron is classified as a lepton (lightweight) and protons and neutrons are classified as baryons (heavyweight). In 1936, the muon was discovered by Carl D. Anderson and Seth Neddermeyer while study cosmic rays. The muon was initially thought to be Yukawa's meson. However, further studies on cosmic rays revealed that another subatomic particle, the pion, was Yukawa's proposed meson, with the muon being classified as a lepton. Cosmic ray research was starting to uncover more subatomic particles than those discovered in the classical era. Over the following decades, further discoveries about particles and their interactions led to the development of the Standard Model, the most successful framework for understanding particle physics.

Developed in the early 1970s, the Standard Model has successfully explained almost all experimental results in particle physics [11]. It argues that almost all the results in

particle physics can be explained by the properties and interactions of a small number of elementary particles. The two basic types of elementary particles are leptons and quarks. There are six leptons, which includes charged particles like the electron and muon, as well as their corresponding neutrinos, which are electrically neutral particles that have very little mass. Quarks, on the other hand, are the building blocks of baryons (protons, neutrons, etc.) and mesons (such as pions). Both leptons and quarks have corresponding antimatter particles, which have the same mass and opposite charge. Interactions between these particles are mediated by force-carrying particles called bosons. An example of a force carrying particle is a photon, which mediates the electromagnetic force. A summary of the Standard Model is shown in figure 2.1.

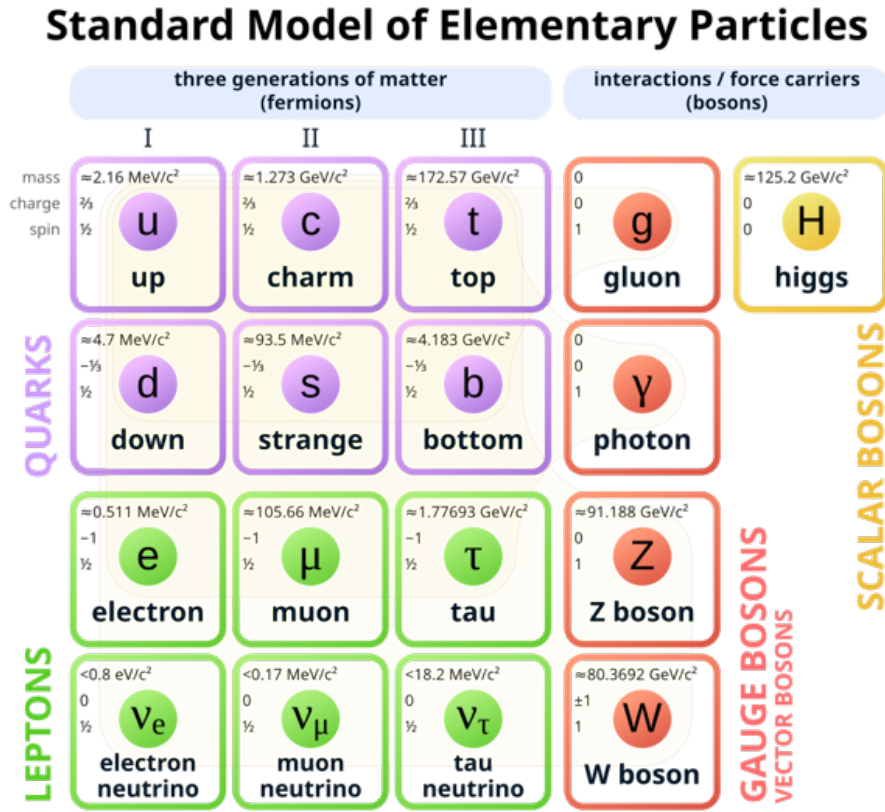


Figure 2.1: The Standard Model [1].

Figure 2.1 shows that leptons and quarks are classified into 3 different generations. All the elements in the periodic table, and thus all stable matter in the universe, are composed of 1st generation particles. For reasons that aren't entirely understood, there exists heavier 2nd and 3rd generation particles that quickly decay into stable particles [12]. This is where the muon fits into the Standard Model. The muon is a 2nd generation lepton.

It is similar to the electron in terms of charge, but its mass is over 200 times greater (105.66 MeV) [7]. A muon has a lifetime of 2.1970 μs before it decays into an electron, an electron antineutrino, and a muon neutrino via the weak interaction [7]. While muons can be produced in particle accelerators, their primary source is cosmic rays.

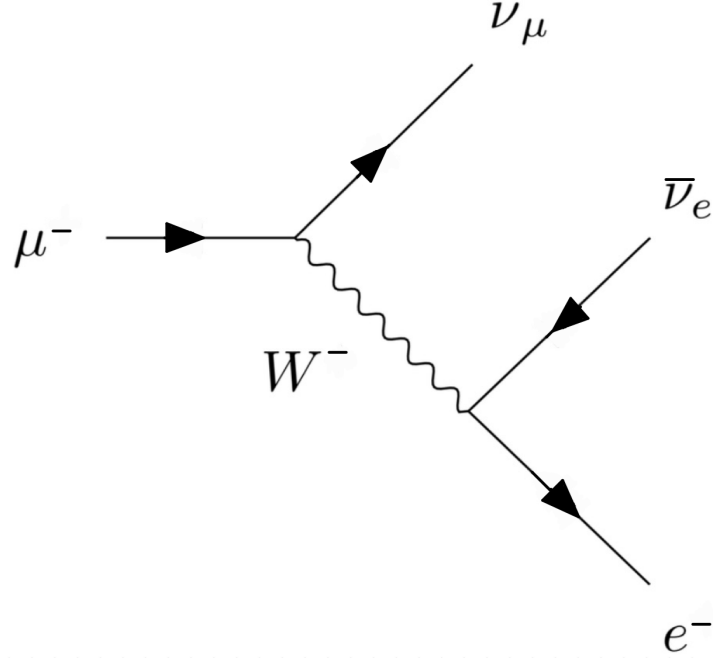


Figure 2.2: Modified diagram showing that muon decay Feynman diagram. Adapted from [2].

2.2 Cosmic Rays

The earth's upper atmosphere is constantly bombarded by high-energy charged particles called primary cosmic rays. Although the exact origin of cosmic rays is not fully understood, the prevailing theory is that some come from distant supernovae [13]. Approximately 98% of cosmic rays are protons or heavier nuclei, and the remaining 2% are electrons [3]. When primary cosmic rays strike a nucleus in Earth's upper atmosphere, typically an oxygen or nitrogen molecule, a cascade of secondary particles are produced. Most of the energy in the interaction is used to create mesons, which consist of a quark and antiquark pair. The most common mesons produced in this interaction are pions (π^+ , π^- , π^0) and kaons (K^+ , K^- , K^0). These mesons have very short lifetimes (approximately 26 ns for charged pions and 12 ns charged kaons [7]), and decay almost immediately

after production. As a result, the mesons themselves do not reach the Earth's surface, but their decay products do. This is where cosmic ray muons come from. The various decay and interaction chains that result from primary cosmic rays striking earths upper atmosphere illustrated in figure 2.3.

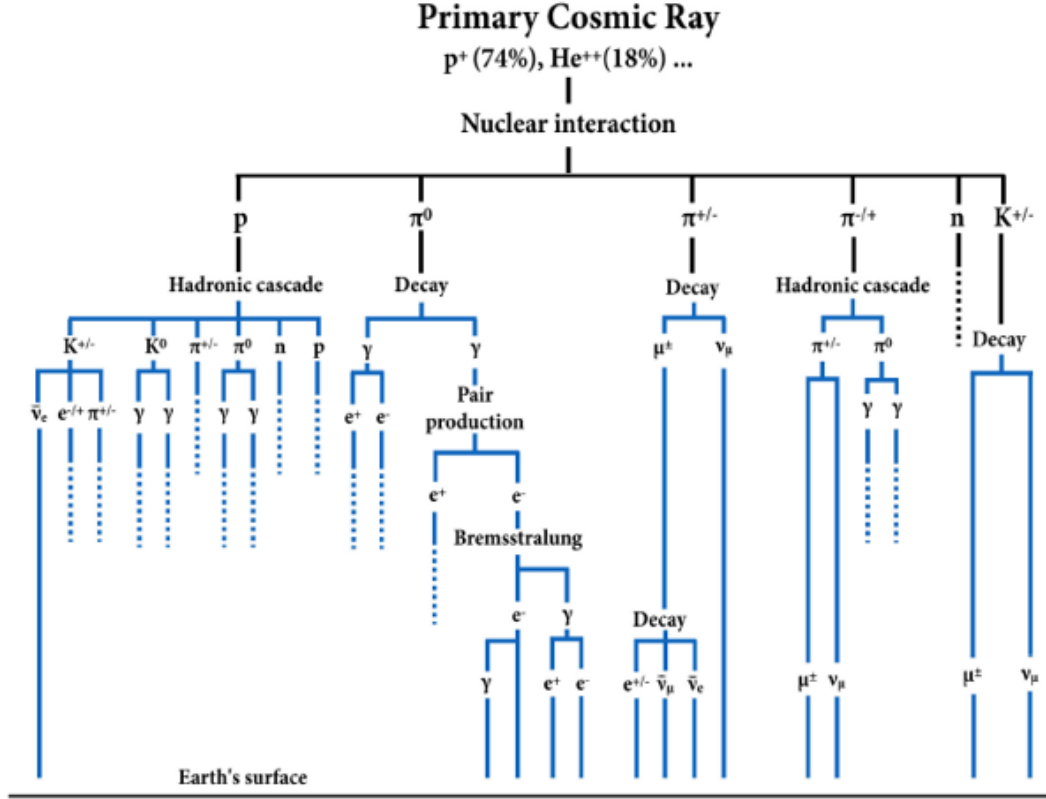


Figure 2.3: An illustration showing the various decay chains as a result of cosmic rays hitting Earth's atomosphere [3].

Of particular interest in this figure are the decay chains that produce muons. One possibility is that charged mesons interact with air molecules, generating additional charged mesons (see the hadronic cascade in Figure 2.3). Both charged pions and charged kaons decay into muons. The decay modes and approximate branching fractions are shown below.

$$\pi^+ \rightarrow \mu^+ + \nu_\mu \quad (99.99\%) \quad (2.1)$$

$$\pi^- \rightarrow \mu^- + \bar{\nu}_\mu \quad (99.99\%) \quad (2.2)$$

$$K^+ \rightarrow \mu^+ + \nu_\mu \quad (63.56\%) \quad (2.3)$$

$$K^- \rightarrow \mu^- + \bar{\nu}_\mu \quad (63.56\%) \quad (2.4)$$

These values are based off [7].

Muons are the most abundant charged particles showering on the Earth's surface. It is estimated that each square centimeter of the Earth is hit by one muon every minute. The detection of muons on the surface of the Earth is direct verification of special relativity, a topic that will be further explained in section 2.3. Muons decay via the weak force into an electron, a neutrino and an antineutrino with a branching fraction of 100% [7].

$$\mu^+ \rightarrow e^+ + \nu_e + \bar{\nu}_\mu \quad (100\%) \quad (2.5)$$

$$\mu^- \rightarrow e^- + \bar{\nu}_e + \nu_\mu \quad (100\%) \quad (2.6)$$

As a result, muons that decay before reaching the Earth's surface produce electrons (and positrons). These particles are important to consider because they are a source of ionizing radiation, which can sometimes be mistaken for muons by muon detectors.

2.3 Special Relativity

Muons are produced around 15 km above the Earth's surface, meaning that the muon would need a decay length of approximately 15 km to reach the surface. If we ignore relativist effects and calculate the distance a muon traveling at the speed of light would cover during its mean lifetime, we obtain the following result.

$$d = c\tau \quad (2.7)$$

$$d = \left(299,792,458 \frac{\text{m}}{\text{s}}\right) \times (2.2 \mu\text{s}) \approx 700 \text{ m} \quad (2.8)$$

However, Einstein's theory of relativity says that times slower for fast-moving particles. Let's consider a muon with an energy of 4 GeV, which is the mean energy of muons at the surface of the earth [3]. The time dilation effect can be calculated by computing the Lorentz factor [14].

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (2.9)$$

$$\gamma = \frac{E}{E_0} \quad (2.10)$$

It is easiest to calculate this using the rest mass and mean energy.

$$\gamma = \frac{4}{0.10565} = 37.86 \quad (2.11)$$

The muon decay measured by a stationary observer on earth can be computed using the time dilation equation [14].

$$t' = \gamma t \quad (2.12)$$

$$t' = 37.86 \times 2.2 \mu\text{s} = 83.29 \mu\text{s} \quad (2.13)$$

Recomputing the decay length using the calculated relativistic lifetime shows a decay length greater than 15 km.

$$d = c\tau_{\text{rel}} = \left(299,792,458 \frac{\text{m}}{\text{s}}\right) \times (83.29 \mu\text{s}) \approx 25 \text{ km} \quad (2.14)$$

Thus, the detection of muons on the Earth's surface is experimental evidence for Einstein's theory of special relativity.

2.4 Muon Interactions with Matter

Muons interact with matter primarily through coulomb forces. Travelling through the atmosphere, muons will interact with many electrons. Over the course of these interactions, the muon will transfer energy to the electrons at its own expense. As the energy of the muon decreases, so does its velocity. This loss of energy is called linear stopping power and is defined as the differential energy loss within the material divided by the differential path length [4].

$$S = -\frac{dE}{dx} \quad (2.15)$$

The Bethe formula is used to describe the specific energy loss for charged particles travelling through matter. It is applicable to particles that have velocities that are large compared to the velocities of the electrons in the absorbing atoms. The Bethe formula [4] is defined below.

$$-\frac{dE}{dx} = \frac{4\pi e^2 z^2}{m_\mu^2 v^2} N Z \left(\ln \left(\frac{2m_\mu v^2}{I} \right) - \ln \left(1 - \frac{v^2}{c^2} \right) - \frac{v^2}{c^2} \right) \quad (2.16)$$

N is the number density of the absorbing atoms, Z is the atomic number of the absorbing atoms, m_μ is the muon rest mass, v is the velocity of the primary particle, ez is the charge of the primary particle, and e is the electron charge.

From the Bethe formula, the estimated energy loss of a muon as it travels a vertical distance H through air with density ρ_{air} is given by the following expression [5].

$$\Delta E = \left(\frac{2 \text{ MeV}}{\text{g cm}^{-2}} \right) \times H \times \rho_{\text{air}} \quad (2.17)$$

From this formula, we can see that muons that travel longer distances through the atmosphere (such as muons coming in at different zenith angles) will lose more energy. Also, different atmospheric conditions that change air density (pressure, temperature) impact the amount of energy the muon loses on its path through the atmosphere. Factors that change the linear stopping power of muons will thus impact the muon flux because only muons that had sufficient energy to reach the detector will be observed.

Chapter 3

Teachspin and Cosmic Watch Muon Detectors

There are two different muon detectors used for this project. The first, the Teachspin muon detector [5], was purchased prior to the start of this project. The second, the Cosmic Watch muon detector, was constructed following the Cosmic Watch project, a guide developed by Dr. Spencer N. Axani and colleagues at MIT [6]. This guide instructs students on how to build their own muon detectors.

3.1 Teachspin Muon Detector

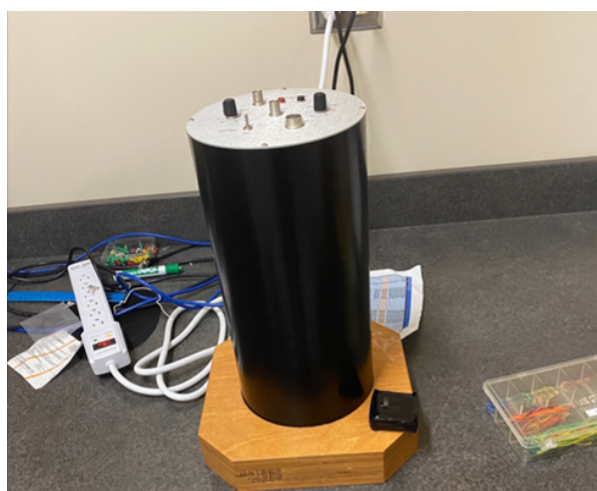


Figure 3.1: Base of the Teachspin muon detector.

Inside the Teachspin muon detector, there is a cylindrical scintillator made of polyvinyltoluene, with a diameter of 15 cm and a height of 12.5 cm. A charged particle passing

through this scintillator will deposit its kinetic energy through ionization, which will promote electrons of molecules within the scintillator to excited states. Upon deexcitation, light will be emitted near the blue portion of the electromagnetic spectrum. Photons are emitted at a rate of approximately one optical photon per 100 eV of energy deposited. Consequently, muons hitting the scintillator will cause photons to be emitted, a signal that can be detected and amplified by a photomultiplier tube [5].

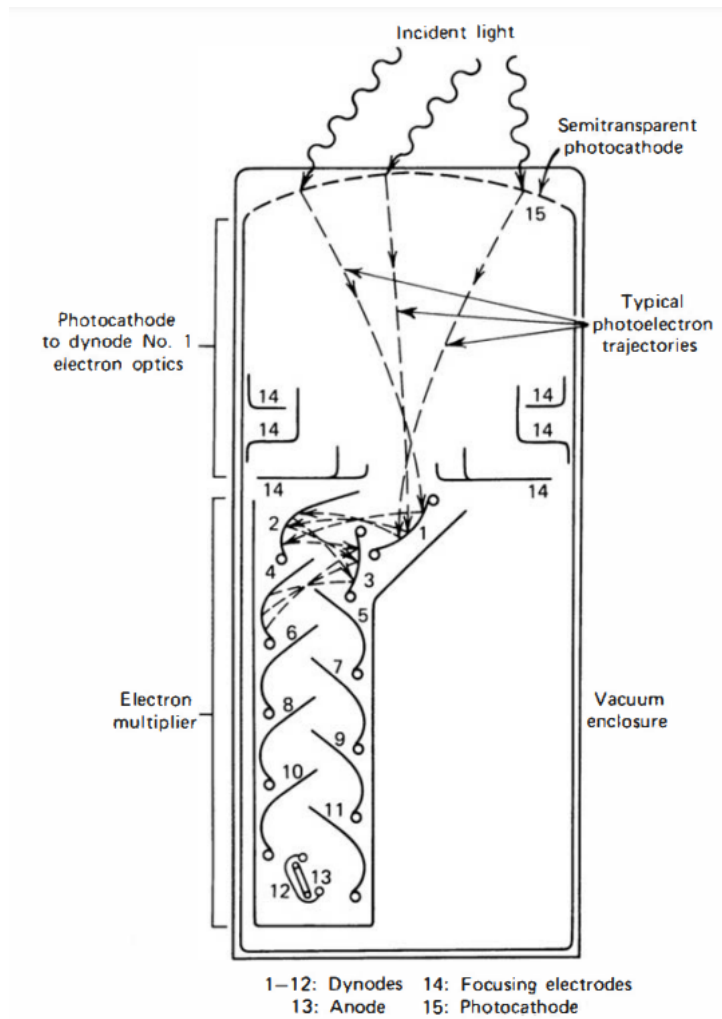


Figure 3.2: Diagram of a PMT [4].

A photomultiplier tube consists of two main components: the photosensitive layer and the electron multiplier. The photosensitive layer is made up of a photocathode, which emits low-energy electrons when it absorbs a photon. As a result, a charged particle hitting the scintillator will release photons that will in turn hit the photocathode and release a few hundred electrons. This pulse would be much too small to produce a sufficient signal, so

an electron multiplier is needed to amplify this signal. It achieves this by using plates of a dynode material, which will release multiple electrons when struck by an electron. By arranging these plates in a particular geometry with a potential difference applied, a few hundred electrons can be amplified to 10^7 to 10^{10} electrons [4]. This large electrical pulse is then strong enough to be analyzed by the detector electronics. This process illustrated in figure 3.2.

The signal from the PMT is amplified by a two-stage amplifier to increase its magnitude. Then, a discriminator is used to filter out events by setting a voltage threshold, which can be set by the control knob. This is used to filter out background events, as it is assumed that the larger signals are a result of muons hitting the scintillator. The last main component is the Field-Programable Gate Array (FPGA), which is used to record muon decay events within the scintillator. Each time a muon hits the scintillator the FPGA starts a precise timer. If a second event is recorded within 20 microseconds, then the detector assumes the second particle is a high-energy electron produced by the muon's decay. This allows for the creation of a decay histogram to measure the muon lifetime. This data is recorded on a computer via a USB cable [5].

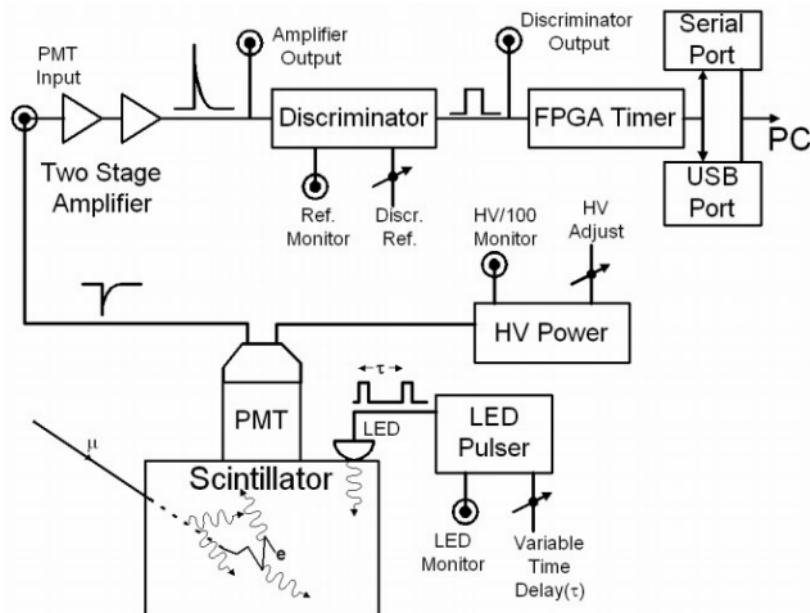


Figure 3.3: Block Diagram for Teachspin Detector Circuit [5]

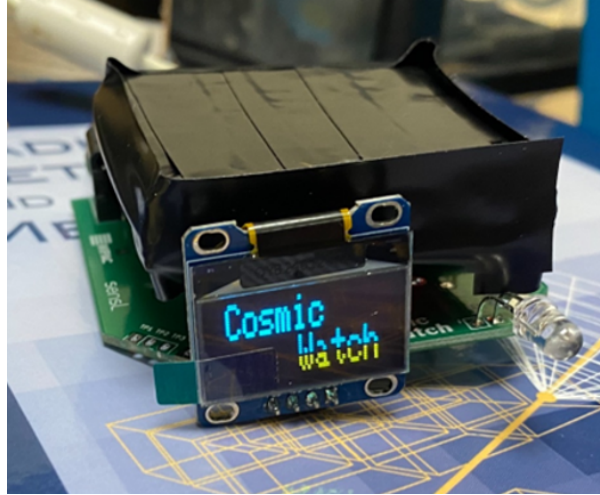


Figure 3.4: Cosmic Watch muon detector.

3.2 The Cosmic Watch Muon Detector

3.2.1 Cosmic Watch Scintillator

Similarly to the Teachspin detector, the Cosmic Watch detector utilizes an organic scintillator (this one is polystyrene-based) to detect ionizing radiation. The scintillator was precut into a $5 \times 5 \times 1 \text{ cm}^3$ rectangular prism and polished to improve optical transparency. Steven MacDonald, StFX University's machinist, drilled four holes in the scintillator to allow for mounting on the detector. It is crucial to drill at a low rotational speed (approximately 60 rpm) when working with plastic scintillators to prevent cracks. The scintillator was then wrapped in reflective foil, leaving a $2 \times 2 \text{ cm}$ hole to couple the scintillator with the Silicon Photomultiplier (SiPM), which detects the photons emitted by the scintillator. The combination of the polished scintillator's increased optical transparency and the reflective foil serves to increase the number of photons incident on the SiPM [6].

3.2.2 The Silicon Photomultiplier

The low cost and compact design of the Cosmic Watch detector are largely attributed to its photon detection method. Many detectors, including the Teachspin muon detector, utilize Photomultiplier Tubes (PMTs) for photon detection. However, PMTs are bulky, require high voltages, and are expensive, making them not ideal for a portable detector like the Cosmic Watch muon detector. Fortunately, advancements in semiconductor technology have led to the development of Silicon Photomultipliers (SiPMs). These devices



Figure 3.5: Scintillator partially wrapped in reflective foil.

operate and low voltages and are compact, making the portable design of the Cosmic Watch detector possible.

SiPMs are made up of densely packed P-type N-type semiconductor junctions forming P-N junctions. In the P-N junction, there is a depletion region filled with bound electrons. When a photon hits the depletion region, an electron is freed and transported to the conduction band. Due to the potential difference applied to the SiPM, these electrons will gain enough energy to release additional electrons to the conduction band, creating an avalanche of millions of electrons. This current will cause silicon to become less conductive and quenching resistors lower the potential difference across the P-N junction, stopping this electron cascade [3].

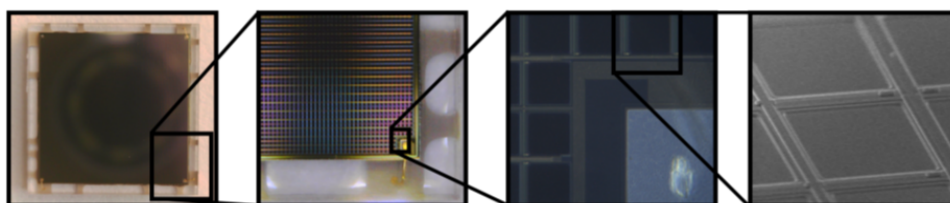


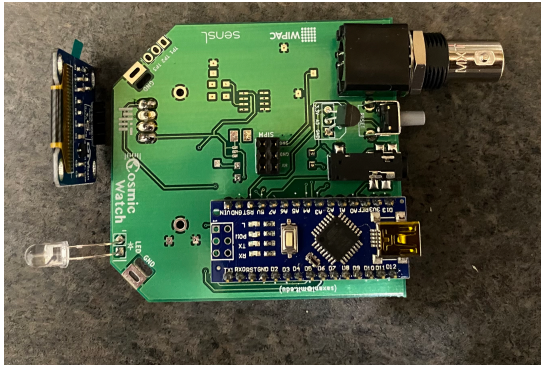
Figure 3.6: Magnified view of the SiPM [3].

A SiPM used for this detector is MicroFC 60035 C-Series SiPM. It contains 18980 P-N junctions, known as microcells. Each microcell allows current to flow when struck by a photon. The SiPM is most sensitive to photons with wavelengths around 450 nm [15], which is close to the emission wavelength of the scintillator (typically 400-420 nm [3]). The total current flowing through the SiPM is proportional to the number of microcells are struck by photons.

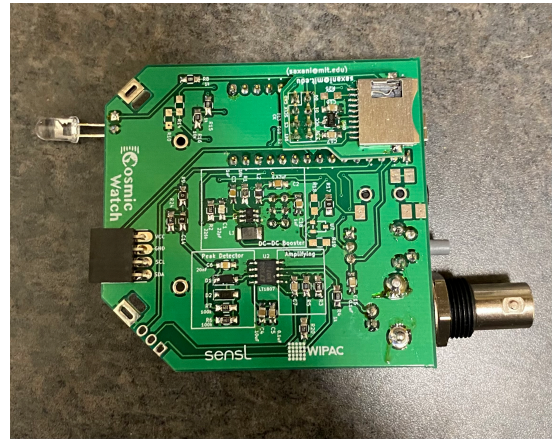
The SiPM was soldered onto a printed circuit board, then coupled with the scintillator. The scintillator-SiPM pair were then wrapped in black electrical tape to prevent outside light from activating the SiPM. When a charge particle hits the scintillator, it emits light proportional to the energy deposited within the scintillator. This SiPM will then draw current at a rate proportional to the number of microcells hit by photons. This will create a voltage plus that can be interpreted by the detector electronics.

3.2.3 Detector Electronics

Printed Circuit Boards (PCBs) as well as other electronic components are used to process the voltage pulses from the SiPM and perform other crucial functions. The PCB acts as the general wiring for the circuit, and the circuit components are soldered to the boards to secure a connection. There are 3 PCBs used for this detector. The first of which, the main PCB, contains the circuitry used to identify muon events and collect data. The second is the SiPM PCB, which connects the SiPM to the main PCB. The last one is the SD card PCB, which attaches an SD card slot so that data can be easily recorded on a SD card. The circuit components were soldered by Craig Seaboyer, StFX university's electronics consultant.



(a) Top side of mounted PCB



(b) Bottom side of mounted PCB

Figure 3.7: Cosmic Watch muon detector main PCB

The full schematic of the detector electronics is shown in Figure 3.8. As described earlier, when a particle strikes the scintillator, it generates a voltage spike in the SiPM. This voltage spike is then amplified by a factor of 24 by the amplification circuit. The voltage peak is further extended to approximately 0.5 ms by the peak detector circuit to ensure

that the peak is long enough to be read by the Arduino [6]. The Arduino is programmed to perform multiple functions, including:

- Setting the voltage threshold for a muon event.
- Measuring the size of the voltage pulses.
- Recording the time of each muon event.
- Tracking detector deadtime.
- Sending data through USB for analysis on a computer.
- Displaying the muon count and flux rate on an OLED screen.
- Activating LED when a muon is detected.

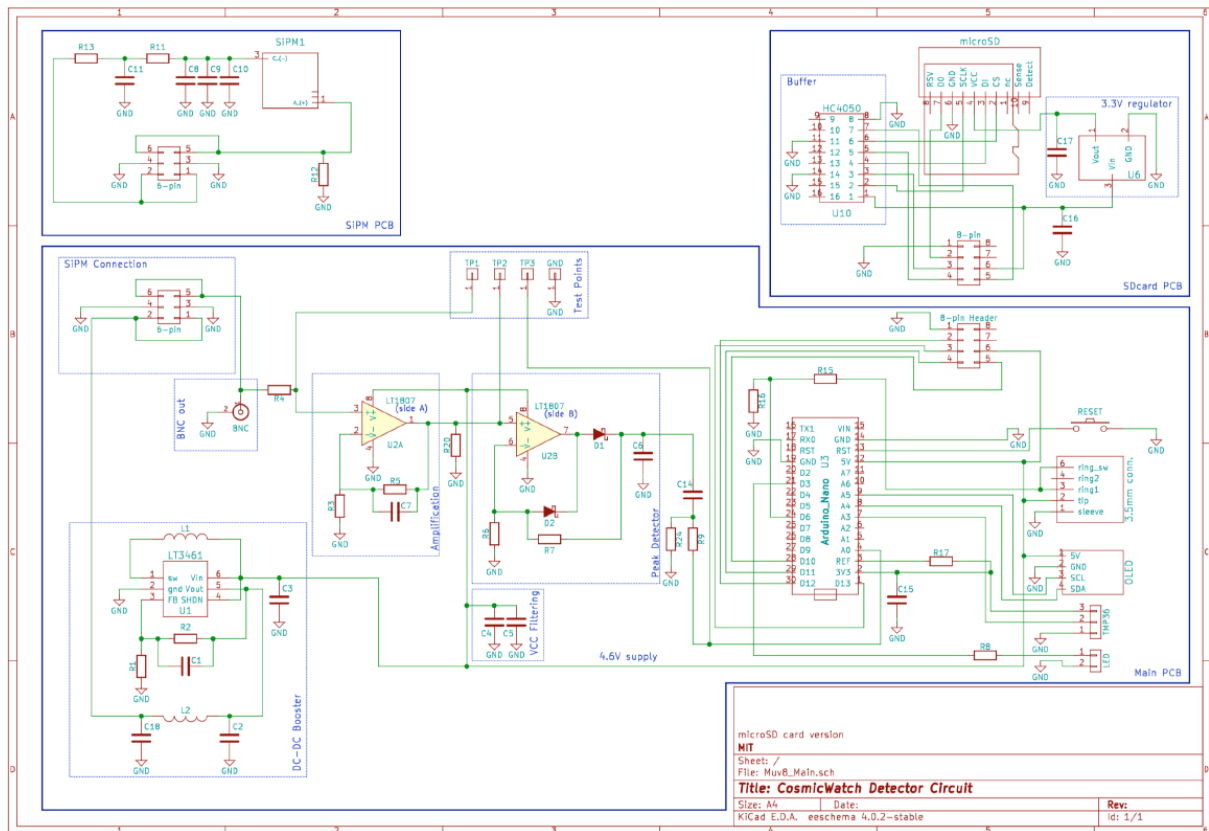


Figure 3.8: Circuit diagram of Cosmic Watch muon detector [6].

Chapter 4

Data Collection and Analysis

4.1 Teachspin Muon Detector

4.1.1 Example Data Interpretation

Data from the Teachspin muon detector is stored in a .data file. This can be easily imported into an Excel document, as shown in figure X. This dataset consists of two fields. The first field contains a positive integer that has multiple interpretations. If the positive integer N is 40000 or greater, then $N - 40000$ corresponds to the number of muons detected during that second. If the positive integer N is less than 40000, it indicates the time, in nanoseconds, between two successive signals, signifying a decay event. The second field measures the number of seconds since January 1, 1970 (00:00:00 1970-01-01 UTC), which is a standard time measurement in computer software.

40003 1717421226
40005 1717421227
40001 1717421228
40003 1717421229
40004 1717421230
40002 1717421231
40003 1717421232
2220 1717421232
40005 1717421233
40004 1717421234
40006 1717421235
40005 1717421236
40002 1717421237

Figure 4.1: Raw data from Teachspin detector

The example data shows us that there were three muons that passed through the scintillator 1717421226 seconds after the 1st of January 1970. It also says that there was a decay event 7 seconds later, with 2220 ns between the 2 pulses. While the data format may seem unconventional, it simplifies the process of performing flux calculations and making a decay histogram to measure muon lifetime.

4.1.2 Calibration of Teachspin Muon Detector

The first task of this project was to properly calibrate the department's Teachspin Muon Detector. The calibration can be adjusted using the HV (High Voltage) and control knob. The HV knob changes the voltage supplied to the PMT (Photomultiplier Tube). When a muon passes through the scintillator, photons are emitted that is proportional to the amount of energy deposited into the scintillator. The PMT is used to amplify and convert this signal into a voltage pulse. Adjusting the HV knob will change the gain and sensitivity of the PMT. The voltage can be measured using a digital multimeter via the HV monitor.

The control knob adjusts the threshold voltage of the discriminator. The threshold is used to reject voltage pulses that are too small, and likely a result of background radiation. The discriminator will reject all pulses that are less than the threshold voltage. Similarly to the HV knob, a digital multimeter can be used to measure the threshold voltage.

Using the recommended calibrations for the control and HV knobs yielded unphysical results (around 1000 per second when we would expect to see closer to 200 per minute). I decided that the best course of action was to attempt to calibrate based on a known physical constant. The Teachspin Muon Detector uses a Field-Programmable Gate Array (FPGA) to determine if a muon decays within the detector. These events will appear as two voltage pulses within a 20 μ s window. By plotting the decay times (on the x-axis) against the number of events (on the y-axis), you can extract the muon lifetime. To analyze this data, the histogram is fit with a curve that combines an exponential decay function (to model the muon decay process) and a constant term (for the events arising from the detection of two successive muons). The exponential component reflects the characteristic decay behavior, while the constant term accounts for background events from consecutive muon decays. Teachspin provides muon physics software to do this analysis, however I found it necessary to write my own python script to analyze my data. This is because Teachspin

uses an outdated algorithm to fit the curve. Using my own code also allows me to quickly compute things like the flux and decay rate, as well as compare multiple trials.

I repeated long trials (multiple days) with different HV and Control voltages, comparing the experimental and theoretical lifetime. We expected a lifetime $2.120 \mu\text{s}$, which is the mean decay time of negative and positive muons [7]. We found that setting the HV to 658 V and the Discriminator voltage to 201mV yielded the expected lifetime and reasonable muon flux. An example of the lifetime plot at these voltages is shown below.

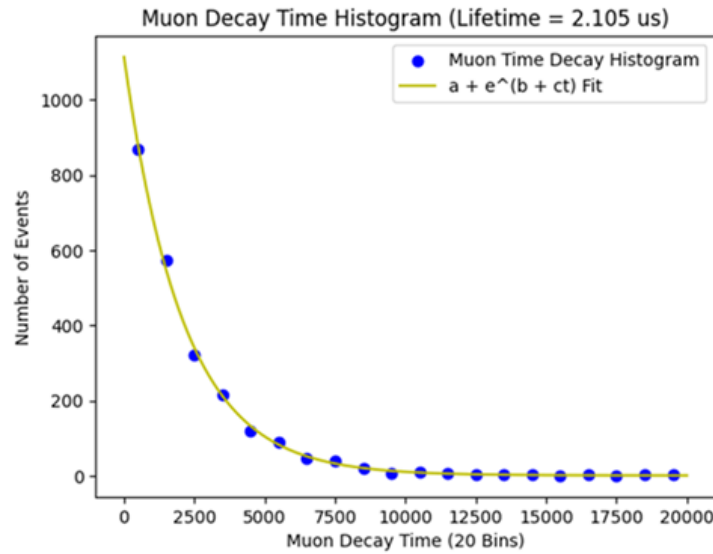


Figure 4.2: Decay histogram for measuring muon lifetime.

After finding a calibration that yields the expected muon lifetime, I started running trials in different rooms on the NFSC. As expected, we saw higher flux levels on the top floor because of higher altitude and less solid material above the detector. We noticed that changing outlets would change the HV and discriminator voltage, so careful attention must be taken to ensure consistency.

4.1.3 Development of New Teachspin Muon Detector Software

Previously, data collection for the muon detector was done using Teachspin's software. However, this software had several limitations, including an outdated curve fitting algorithm. Additionally, it was only compatible with computers running Windows 10 or earlier, which posed a problem for future outdoor experiments, as my laptop does not support Windows 10. To address these issues, I decided to develop new software for data collection that would be compatible with a broader range of devices.

I began by printing it the serial data coming in from the detector, which outputted strings. Through analysis, I was able to identify which strings corresponded to muons hitting the detector and which indicated particle decays. Encoded in these strings was the muon rate and the decay times. Together with Matt Penner, I developed software that updates a text file in real time with this detector data. This data is in the same format as the Teachspin software data, so the code previously used for data analysis is compatible with the new software's data

In addition, the new software automatically generates two graphs. The first graph tracks the mean muon flux over the entire run, plotted against time. The second graph calculates the current muon flux and updates every 10 minutes, displaying both the mean flux and its standard deviation. Example plots are displayed below.

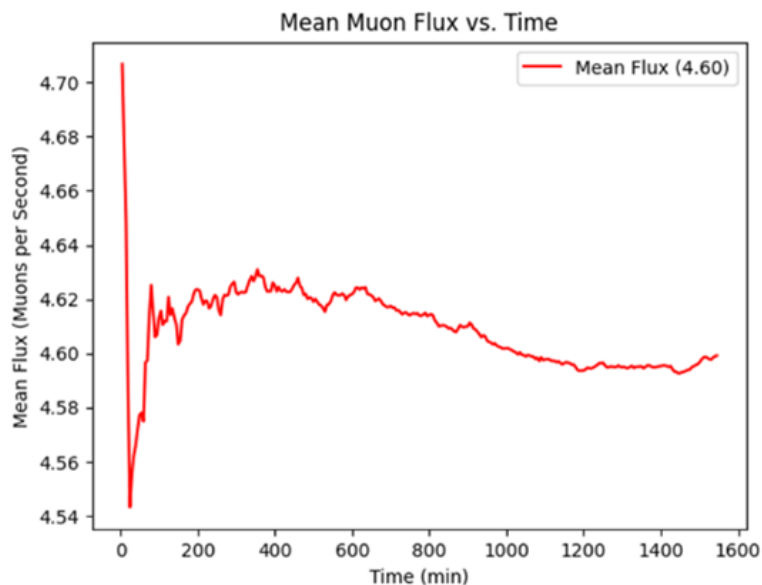


Figure 4.3: Mean muon flux plot

The software is written in python so that any computer with a python interpreter can collect flux data of the Teachspin muon detector. I have included this code in Appendix A so that it can be used for future research.

4.1.4 Inconsistencies with Battery Trials

In preparation for a muon flux experiment at Keppoch Mountain, trial were my laptop and an AC power source to match the experiment conditions that would be used on the mountain. While some trials would match up with the experiments using the wall outlet

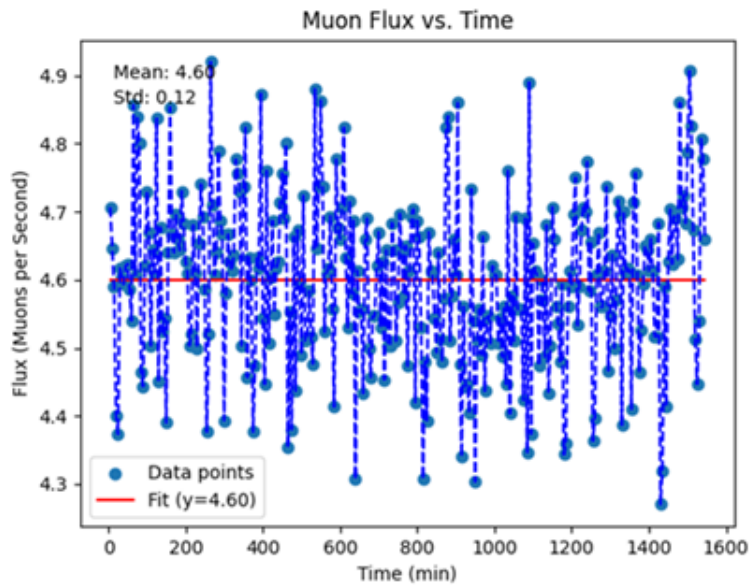


Figure 4.4: Current muon flux (updated every 10 minutes).

of the power source, other trials would yield unphysical results. It seems that something as small as moving the battery mid trial or adjusting the cable would affect the results. For some of the longer trials, the muon flux would change greatly throughout the trial. This issue persisted when switching cables, which suggests the issue might be the battery.

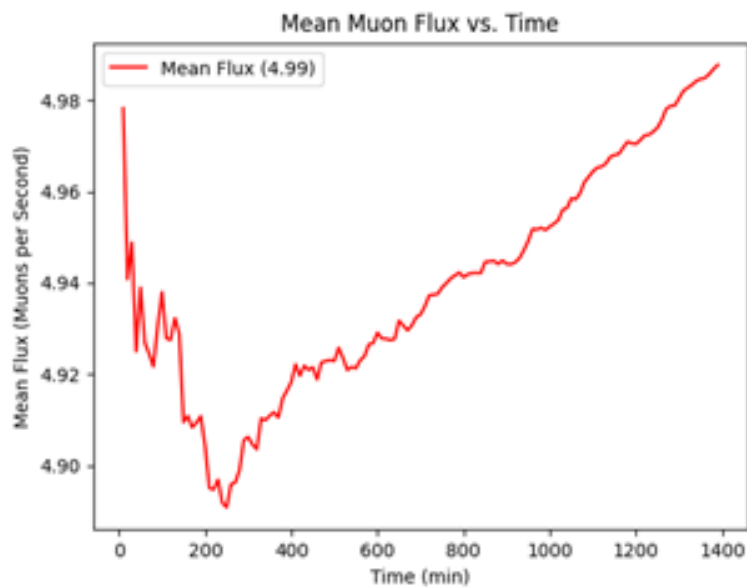


Figure 4.5: Inconsistent flux from battery trial.

4.2 Cosmic Watch Muon Detector

4.2.1 Example Data Interpretation

```
#####
### CosmicWatch: The Desktop Muon Detector
### Comp_date Comp_time Event Ardn_time[ms] ADC[0-1023] SiPM[mV] Deadtime[ms] Temp[C] Name
#####
Device ID(s): Muon_STFX_1
2024-07-22 12:17:44.180971 1 184 120 23.58 76 22.94 Muon_STFX_1
2024-07-22 12:17:44.361049 2 366 89 21.05 80 22.62 Muon_STFX_1
2024-07-22 12:17:46.959594 3 2959 399 87.30 154 22.72 Muon_STFX_1
2024-07-22 12:17:47.719950 4 3721 96 21.67 194 22.94 Muon_STFX_1
2024-07-22 12:17:48.310769 5 4311 156 27.28 233 22.94 Muon_STFX_1
2024-07-22 12:17:50.856618 6 6857 49 15.48 307 22.72 Muon_STFX_1
2024-07-22 12:17:51.591457 7 7592 56 16.76 345 22.72 Muon_STFX_1
2024-07-22 12:17:54.351306 8 10350 88 20.96 451 22.94 Muon_STFX_1
2024-07-22 12:17:54.675394 9 10674 96 21.67 455 22.94 Muon_STFX_1
2024-07-22 12:17:55.738407 10 11735 96 21.67 494 22.94 Muon_STFX_1
2024-07-22 12:17:56.399906 11 12394 496 144.49 533 22.51 Muon_STFX_1
2024-07-22 12:17:57.189661 12 13186 114 23.10 576 22.72 Muon_STFX_1
2024-07-22 12:17:58.123431 13 14120 56 16.76 615 22.94 Muon_STFX_1
2024-07-22 12:17:59.157247 14 15153 51 15.86 654 22.62 Muon_STFX_1
2024-07-22 12:17:59.570763 15 15567 88 20.96 658 22.62 Muon_STFX_1
```

Figure 4.6: Cosmic Watch muon detector example data.

Here, I have included some sample data (Figure 4.6) from one of the preliminary Cosmic Watch Detector trials. This data can be collected using any computer with a python interpreter using the code in Appendix B. Below is a breakdown of what each column measures [6]:

- **Comp_date/Comp_time:** The date and time when the computer recorded the data.
- **Event:** The number of muons measured since the recording started. It corresponds to each charged particle that has produced a voltage pulse higher than the threshold. The threshold can be adjusted in the Arduino code.
- **Ardn_time:** The time in milliseconds since the measurement started.
- **ADC:** A number between 0 and 1023, representing the size of the voltage pulse.
- **SiPM:** The SiPM voltage in millivolts (mV), calculated using the ADC value.
- **Deadtime:** The cumulative deadtime of the detector, which is the time when the Arduino could not detect a muon. This value must be subtracted from the Arduino time when performing flux calculations.
- **Temp:** The temperature in Celsius measured by the temperature sensor on the detector.

4.2.2 Flux Measurements and Error Bars

After building the first cosmic watch detector, I decided to test its performance by measuring the muon flux on two different floors of a building on campus. Based on the discussion of muons interactions with matter, it is expected that muons that travel through more air and solid material have a higher probability of slowing down and decaying. Therefore, a detector on the 3rd floor should register fewer muons compared to one on the 4th floor, as the muons must pass through more dense material to reach the 3rd floor. The graph below displays the muon flux difference between the 3rd and 4th floor of Mulroney Hall at StFX University. The flux rate appears to stabilize after about an hour. This is because as the detector records more events, the mean should converge to a more accurate value for flux.

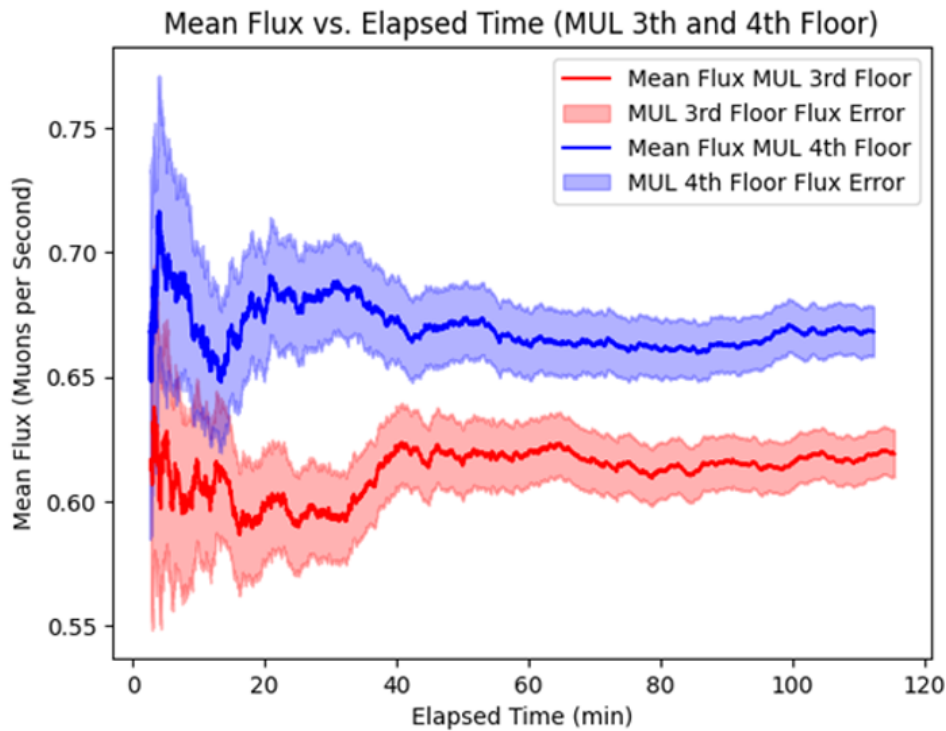


Figure 4.7: Difference in mean muon flux between 3rd and 4th floor of Mulroney Hall.

The shaded region represents all values within one standard deviation of the mean. Muon flux follows a Poisson distribution, which is a statistical distribution where events occur at a constant mean rate, independent of the time since the last event. In the Poisson distribution, the standard deviation is the square root of the number of events that have occurred [16]. For example, if 100 events occur within 100 seconds, the flux would be 1

muon per second (i.e., $\frac{100}{100}$). The standard deviation, in this case, would be $\frac{\sqrt{100}}{100} = 0.1$.

4.2.3 Coincidence Measurements

One of the major advantages of the Cosmic Watch muon detector is the ability to link multiple detectors in coincidence mode. This is done by connecting both detectors with a 3.5 mm male-to-male audio cable and resetting both detectors. In coincidence mode, one detector operates normally while the second detector monitors for muons but only triggers when the first detector detects a muon at the same time. This is very effective at limiting background radiation. For example, it is unlikely that a beta particle would deposit enough energy to trigger both detectors. According to Dr. Spencer N. Axani's Cosmic Watch paper, based on the time sensitivity of the Arduino and the average muon flux, only about 1 in every 15,000 events results from both detectors triggering by chance [3]. As a result, I constructed two Cosmic Watch muon detectors so that altitude experiments can be preformed using coincidence measurements.

4.2.4 Keppoch Experiment

At the end of the summer, I took the Cosmic Watch detectors to the top of Keppoch mountain (approximately 220m above sea level) and saw slightly high flux rates at the top of Keppoch. The detectors were run for about 2 hours, and we recorded a 9% flux difference between Keppoch and a reference trial at 70m elevation. While this is good for a preliminary trial, we ideally would like much longer trials at varying altitudes to mitigate the effect of other variables that affect muon flux. The graph comparing these two trials in in the figure below. Note that the flux rates are lower in these trials compares to the Mulroney trials as the detectors are in coincidence mode.

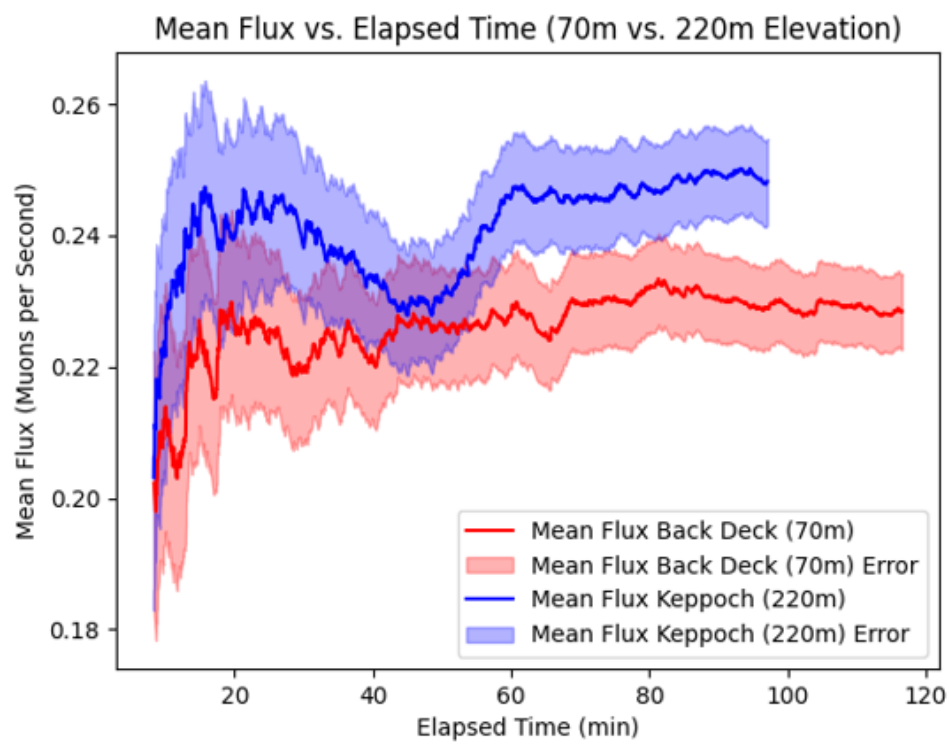


Figure 4.8: Difference in muon flux between Keppoch mountain (220m) and reference point (70m)

Chapter 5

Conclusion and Future Work

In conclusion, this project has provided valuable insights into the detection of cosmic ray muons using the Teachspin and Cosmic Watch muon detectors. By calibrating the Teachspin muon detector and constructing the Cosmic Watch detector, I have gained practical experience in particle detection and flux experiments. The new Teachspin muon detector software developed for this project makes the detector more accessible and enhances its utility, which will be an asset in future experiments. Initial trials at Keppoch Mountain showed an approximately 9% difference in the flux between 70m and 220m. However, given that these trials were only 2 hours long, further data collection is needed to draw conclusive results.

The progress made with both detectors lays the groundwork for future research avenues. With the Teachspin detector, I hope that the calibration technique and new software will aid in future work. The main issue identified during the project was the inconsistencies in battery trials. Further testing with alternative batteries is necessary to determine if these inconsistencies are a result of battery used for the experiments. Once this issue is resolved, additional altitude experiments with both the Teachspin and Cosmic Watch detectors will be conducted to gather more conclusive data.

The newly constructed Cosmic Watch detector allows for many experiments previously not feasible with the Teachspin detector. In addition to the altitude experiments, other experiments for future research have been identified:

- **Angular Distribution of Muons:** By placing two detectors in coincidence mode and adjusting their configurations, it will be possible to isolate for muon flux at specific angles. With this, data can be collected on the angular distribution of

muons.

- **Muon Flux through Different Materials:** Using the two detectors in coincidence mode and placing them on opposing sides of a material, we can investigate the muon flux passing through that material. This could have potential applications for muon tomography, an imaging technique that uses muons.
- **Detector Voltage Distribution:** The Cosmic Watch detector records the magnitude of every voltage pulse caused by a particle hitting the detector. It could be beneficial to see how this distribution changes at different altitudes, where the muons will have different energies. Additionally, observing the distribution under different radiation exposures could further enhance our understanding of detector sensitivity.
- **Investigate Detector Sensitivity to Background Radiation in Coincidence Mode:** By comparing the muon flux detected in coincidence mode with and without exposure to a radiation source, we can assess the effectiveness of coincidence mode in filtering out background radiation. This experiment is relatively simple to set up, and I hope to test it soon.

Furthermore, Dr. Spencer N. Axani and his colleagues have outlined additional potential experiments in their work on the Cosmic Watch detector, which could serve as a valuable resource for future research [3].

Overall, the progress made in this project has not only contributed to a better understanding of muon detection and flux measurements but has also opened up new possibilities for future experiments.

Bibliography

- [1] F. A. Buot, G. T. Maglasang, and A. R. B. Elnar, “Unification of mixed hilbert-space representations in condensed matter physics and quantum field theory,” 2022. [Online]. Available: <https://arxiv.org/abs/2204.07691>
- [2] G. Lukicov, “Alignment of the straw tracking detectors for the fermilab muon $g - 2$ experiment and systematic studies for a muon electric dipole moment measurement,” Ph.D. dissertation, University College London, 2020.
- [3] S. N. Axani, “The physics behind the cosmicwatch desktop muon detectors,” 2019. [Online]. Available: <https://arxiv.org/abs/1908.00146>
- [4] G. F. Knoll, *Radiation Detection and Measurment*, 4th ed. John Wiley Sons, 2010.
- [5] T. E. Coan and J. Ye, *Muon Physics*, TeachSpin Inc., manual for the TeachSpin Muon Detector.
- [6] S. Axani, K. Frankiewicz, and J. Conrad, “The cosmicwatch desktop muon detector: a self-contained, pocket sized particle detector,” *Journal of Instrumentation*, vol. 13, no. 03, p. P03019–P03019, Mar. 2018. [Online]. Available: <http://dx.doi.org/10.1088/1748-0221/13/03/P03019>
- [7] S. Navas *et al.*, “Review of particle physics,” *Phys. Rev. D*, vol. 110, no. 3, p. 030001, 2024.
- [8] D. Griffiths, *Introduction to Elementary Particles*, 2nd ed. John Wiley & Sons, 2008.
- [9] S. Berryman, “Democritus,” in *The Stanford Encyclopedia of Philosophy*, spring 2023 ed., E. N. Zalta and U. Nodelman, Eds. Metaphysics Research Lab, Stanford University, 2023.

- [10] T. E. of Encyclopaedia Britannica. (2023, Apr.) Thomson atomic model. Accessed: 2025-02-17. [Online]. Available: <https://www.britannica.com/science/Thomson-atomic-model>
- [11] CERN. (n.d.) The standard model. Accessed: 2025-02-20. [Online]. Available: <https://home.cern/science/physics/standard-model>
- [12] B. R. Martin, *Nuclear and Particle Physics*. John Wiley & Sons, 2006.
- [13] L. O. Drury, “Origin of cosmic rays,” *Astroparticle Physics*, vol. 39-40, pp. 52–60, 2012, cosmic Rays Topical Issue. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S092765051200045X>
- [14] D. J. Griffiths, *Introduction to Electrodynamics*, 4th ed. Pearson, 2017.
- [15] Onsemi, “Microc series sipm (silicon photomultiplier) datasheet,” n.d., accessed: 2025-03-17. [Online]. Available: <https://www.onsemi.com/pdf/datasheet/microc-series-d.pdf>
- [16] S. M. Ross, “Chapter 5 - special random variables,” in *Introduction to Probability and Statistics for Engineers and Scientists (Fifth Edition)*, fifth edition ed., S. M. Ross, Ed. Boston: Academic Press, 2014, pp. 141–205. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123948113500058>
- [17] S. Axani, “Cosmicwatch desktop muon detector v2,” 2023, accessed: 2025-02-28. [Online]. Available: <https://github.com/spenceraxani/CosmicWatch-Desktop-Muon-Detector-v2>

Appendix A

Python Code for Teachspin Detector

This code was developed to record data from the Teachspin muon detector, providing an alternative to the software supplied with the device, which is limited to Windows 10 or earlier. Unlike the original software, this updated version is compatible with any system that has a Python interpreter installed. This code processes data in the same format as the Teachspin software and automatically generates mean flux plots like those shown in figures 4.3 and 4.4.

```
1 import serial
2 import time
3 import keyboard
4 import matplotlib.pyplot as plt
5 from scipy.stats
6 import linregress
7 import numpy as np
8 import statistics
9
10 #import matplotlib.pyplot
11 # Configure the serial port
12 ser = serial.Serial(
13     port='COM4', # Replace with your USB port
14     baudrate=9600, # Set baud rate as per your device
15     parity=serial.PARITY_NONE,
16     stopbits=serial.STOPBITS_ONE,
17     bytesize=serial.EIGHTBITS,
```

```

18     timeout=5                # Read timeout in seconds
19 )
20
21 # Check if the serial port is open
22 if ser.is_open:
23     print(f"Connected to {ser.port}")
24 time_for_plot = 0
25 plotRate = 300
26 buffer = b''
27 data_list = []
28 decay_list = []
29 time_list = []
30 flux_list = []
31 mean_flux_list = []
32 fluxCount = 0
33 last_time = time.time()
34 last_time2 = time.time()    # Initialize the timer
35
36
37 def append_to_file(filename, data1, data2, current_time):
38     with open(filename, 'a') as file:
39         global fluxCount
40         # Write the number of muons in the last second
41         muon_count = len(data1) + 40000
42         fluxCount += len(data1)
43         file.write(f"{muon_count}_{int(current_time)}\n")
44         # Write the decays
45         for line in data2:
46             if isinstance(line, bytes):
47                 fluxCount += 1
48                 value = int(line, 16) * 20
49                 file.write(f"{value}_{int(current_time)}\n")
50
51 try:

```

```

52     while True:
53         data = ser.read(ser.in_waiting)
54         if data:
55             buffer += data # Append new data to the buffer
56             if b'\r\n' in buffer:
57                 messages = buffer.split(b'\r\n')
58                 # Split the buffer into messages
59                 for message in messages[:-1]:
60                     # Process all complete messages
61                     data_list.append(message)
62                     # Add the raw message to the list
63                     if str(message) != "b'44D'":
64                         decay_list.append(message)
65                 buffer = messages[-1]
66                 # Keep any partial message for the next read
67
68             # Append "s" to the data list every second
69             current_time = time.time()
70             current_time2 = time.time()
71             if current_time - last_time >= 1:
72                 append_to_file('all_data.txt', data_list, decay_list,
73                               last_time)
74                 data_list.clear()
75                 decay_list.clear()
76                 last_time = current_time
77             if current_time2 - last_time2 >= plotRate:
78                 time_for_plot += plotRate / 60
79                 time_list.append(time_for_plot)
80                 flux_list.append(fluxCount / plotRate)
81                 mean_flux_list.append(statistics.mean(flux_list))
82                 fluxCount = 0
83
84                 # Fit a constant line
85                 mean_flux = np.mean(flux_list)

```

```

85     y_fit = [mean_flux] * len(time_list)
86
87     # Calculate statistics
88     mean_flux = np.mean(flux_list)
89     std_flux = np.std(flux_list)
90
91     # Plot points and fit line
92     plt.scatter(time_list, flux_list, label='Data_points',
93                )
94     plt.plot(time_list, y_fit, color='red',
95             label=f'Fit (y={mean_flux:.2f})')
96     plt.plot(time_list, flux_list, color='blue',
97             linestyle='--')
98
99     # Display statistics
100    plt.text(0.05, 0.95, f'Mean: {mean_flux:.2f}',
101            transform=plt.gca().transAxes, verticalalignment='top')
102    plt.text(0.05, 0.90, f'Std: {std_flux:.2f}',
103            transform=plt.gca().transAxes, verticalalignment='top')
104
105    plt.xlabel('Time (min)')
106    plt.ylabel('Flux (Muons per Second)')
107    plt.title("Muon Flux vs. Time")
108    plt.legend()
109    plt.savefig("flux.png")
110    plt.close()
111
112    plt.xlabel('Time (min)')
113    plt.ylabel('Mean Flux (Muons per Second)')
114    plt.title("Mean Muon Flux vs. Time")
115    plt.plot(time_list, mean_flux_list, color='red',
116            label=f'Mean Flux ({mean_flux:.2f})')

```

```
115         plt.legend()
116         plt.savefig("meanflux.png")
117         plt.close()
118
119         last_time2 = current_time2
120
121
122 except KeyboardInterrupt:
123     print("Program interrupted. Closing...")
124 finally:
125     ser.close() # Close the serial port
126     print("Serial port closed.")
```

Appendix B

Python Code for Cosmic Watch Detector

This code is designed for data collection on the Cosmic Watch detector. It was originally written as part of the Cosmic Watch project and is provided here for reference [17].

```
1 import time
2 import glob
3 import sys
4 import os
5 import os.path
6 import signal
7 from datetime import datetime
8 from multiprocessing import Process
9
10 #import numpy as np
11 import tornado.httpserver
12 import tornado.websocket
13 import tornado.ioloop
14 import tornado.web
15 import socket
16 import multiprocessing
17 import math
18 import random
19 import _thread
```

```

20 import serial
21
22 '''
23 This is a Websocket server that forwards signals from the
24     detector to any client connected.
25 It requires Tornado python library to work properly.
26 Please run 'pip install tornado' with python of version 2.7.9 or
27     greater to install tornado.
28 Run it with 'python detector-server.py'
29 Written by Pawel Przewlocki (pawel.przewlocki@ncbj.gov.pl).
30 Based on http://fabacademy.org/archives/2015/doc/WebSocketConsole.html
31 '''
32
33 global ComPort
34 ComPort = None
35
36 global file
37 file = None
38
39
40 def print_help1():
41     print('\n=====HELP=====')
42     print('This code looks through the serial ports.')
43     print('You can select multiple ports with by separating the port number with commas.')
44     print('You must select which port contains the Arduino.\n')
45     print('If you have problems, check the following:')
46     print('1. Is your Arduino connected to the serial USB port?\n')
47     print('2. Check that you have the correct drivers installed:\n')
48     print('\tMacOS: CH340g driver (try: https://github.com/adrianmihalko/ch340g-ch34g-ch34x-mac-os-x-driver)')
49     print('\tWindows: no driver needed')
50     print('\tLinux: no driver needed')

```

```

47
48
49 clients = [] ## list of clients connected
50 queue = multiprocessing.Queue() #queue for events forwarded from
    the device
51
52 class DataCollectionProcess(multiprocessing.Process):
53     def __init__(self, queue):
54         #multiprocessing.Process.__init__(self)
55         self.queue = queue
56         self.comport = serial.Serial(port_name_list[0]) # open
            the COM Port
57         self.comport.baudrate = 9600                # set Baud rate
58         self.comport.bytesize = 8                    # Number of data
            bits = 8
59         self.comport.parity = 'N'                    # No parity
60         self.comport.stopbits = 1
61
62     def close(self):
63         self.comport.close()
64
65     def nextTime(self, rate):
66         return -math.log(1.0 - random.random()) / rate
67
68 def RUN(bg):
69     print('Running...')
70     while True:
71         data = bg.comport.readline()
72         bg.queue.put(str(datetime.now())+"␣"+data)
73
74 class WSHandler(tornado.websocket.WebSocketHandler):
75     def __init__(self, application, request, **kwargs):
76         super(WSHandler, self).__init__(application, request, **
            kwargs)

```

```

77         self.sending = False
78
79     def open(self):
80         print('New_connection_opened_from_' + self.request.
81               remote_ip)
82         clients.append(self)
83         print('%d_clients_connected' % len(clients))
84
85     def on_message(self, message):
86         print('message_received:_%s' % message)
87         if message == 'StartData':
88             self.sending = True
89         if message == 'StopData':
90             self.sending = False
91
92     def on_close(self):
93         self.sending = False
94         clients.remove(self)
95         print('Connection_closed_from_' + self.request.remote_ip)
96         print('%d_clients_connected' % len(clients))
97
98     def check_origin(self, origin):
99         return True
100
101 def checkQueue():
102     while not queue.empty():
103         message = queue.get()
104         ##sys.stdout.write('#')
105         for client in clients:
106             if client.sending:
107                 client.write_message(message)
108
109 def signal_handler(signal, frame):

```

```

110     print('You pressed Ctrl+C!')
111     if ComPort:
112         ComPort.close()
113     if file:
114         file.close()
115     sys.exit(0)
116
117 def serial_ports():
118     """ Lists serial port names
119
120     :raises EnvironmentError:
121         On unsupported or unknown platforms
122     :returns:
123         A list of the serial ports available on the system
124     """
125     if sys.platform.startswith('win'):
126         ports = ['COM%s' % (i + 1) for i in range(256)]
127     elif sys.platform.startswith('linux') or sys.platform.
128         startswith('cygwin'):
129         # this excludes your current terminal "/dev/tty"
130         ports = glob.glob('/dev/tty[A-Za-z]*')
131     elif sys.platform.startswith('darwin'):
132         ports = glob.glob('/dev/tty.*')
133     else:
134         raise EnvironmentError('Unsupported platform')
135     sys.exit(0)
136     result = []
137     for port in ports:
138         try:
139             s = serial.Serial(port)
140             s.close()
141             result.append(port)
142         except (OSError, serial.SerialException):
143             pass

```



```

176     print('['+str(i+1)+']_ ' + str(port_list[i]))
177 print('[_h]_help\n')
178
179 ArduinoPort = input("Selected_Arduino_port:_")
180
181 ArduinoPort = ArduinoPort.split(',')
182 nDetectors = len(ArduinoPort)
183
184 if mode in [2,3,4]:
185     if len(ArduinoPort) > 1:
186         print('---_Error_---')
187         print('You_selected_multiple_detectors.')
188         print('This_options_is_only_compatible_when_
189             recording_to_the_computer.')
190         print('Exiting...')
191         sys.exit()
192
193 port_name_list = []
194
195 for i in range(len(ArduinoPort)):
196     port_name_list.append(str(port_list[int(ArduinoPort[i])
197         -1]))
198
199 if ArduinoPort == 'h':
200     print_help1()
201     sys.exit()
202
203 print("The_selected_port(s)_is(are):_")
204 for i in range(nDetectors):
205     print('\t['+str(ArduinoPort[i])+']_ ' + port_name_list[i])
206
207 if mode == 1:
208     cwd = os.getcwd()

```

```

208     fname = input("Enter file name (default: "+cwd+"/CW_data.
        txt):")
209
210     detector_name_list = []
211
212     if fname == '':
213         fname = cwd+"/CW_data.txt"
214
215     print('Saving data to: '+fname)
216
217     #ComPort_list = np.ones(nDetectors)
218     for i in range(nDetectors):
219         signal.signal(signal.SIGINT, signal_handler)
220         globals()['Det%s' % str(i)] = serial.Serial(str(
            port_name_list[i]))
221         globals()['Det%s' % str(i)].baudrate = 9600
222         globals()['Det%s' % str(i)].bytesize = 8
223         # Number of data bits = 8
224         globals()['Det%s' % str(i)].parity = 'N'
225         # No parity
226         globals()['Det%s' % str(i)].stopbits = 1
227
228         time.sleep(1)
229         #globals()['Det%s' % str(i)].write('write')
230
231         counter = 0
232
233         header1 = globals()['Det%s' % str(i)].readline()
234         # Wait and read data
235         if 'SD initialization failed' in header1.decode():
236             :
237             print('...SDCard.ino detected.')
238             print('...SDcard initialization failed.')

```

```

235         # This happens if the SDCard.ino is
                uploaded but it doesn't see an sdcard.
236         header1a = globals()['Det%s' % str(i)].
                    readline()
237         header1 = globals()['Det%s' % str(i)].
                    readline()
238         if 'CosmicWatchDetector' in header1.decode():
239             print('...SDCard.ino_code_detected.')
240             print('...SDcard_intialized_correctly.')
241             # This happens if the SDCar.ino is
                uploaded and it sees an sdcard.
242             header1a = globals()['Det%s' % str(i)].
                        readline()
243             globals()['Det%s' % str(i)].write('write'
                )
244             header1b = globals()['Det%s' % str(i)].
                        readline()
245             header1 = globals()['Det%s' % str(i)].
                        readline()
246             #header1 = globals()['Det%s' % str(i)].
                readline()
247             header2 = globals()['Det%s' % str(i)].readline()
                # Wait and read data
248             header3 = globals()['Det%s' % str(i)].readline()
                # Wait and read data
249             header4 = globals()['Det%s' % str(i)].readline()
                # Wait and read data
250             header5 = globals()['Det%s' % str(i)].readline()
                # Wait and read data
251
252             det_name = globals()['Det%s' % str(i)].readline()
                        .decode().strip()
253             #print(det_name)
254             if 'Device_ID:' in det_name:

```

```

255         det_name = det_name.split('Device_ID:')
                [-1]
256         detector_name_list.append(det_name)      # Wait and
                read data
257
258
259         file = open(fname, "w")
260         file.write(header1.decode())
261         file.write(header2.decode())
262         file.write(header3.decode())
263         file.write(header4.decode())
264         file.write(header5.decode())
265
266         string_of_names = ''
267         print("\n--_Detector_Names_--")
268         #print(detector_name_list)
269         for i in range(len(detector_name_list)):
270             print(detector_name_list[i])
271             if '\xff' in detector_name_list[i] or '?' in
                detector_name_list[i] :
272                 print('---_Error_---')
273                 print('You_should_name_your_CosmicWatch_
                Detector_first.')
274                 print('Simply_change_the_DetName_variable
                in_the_Naming.ino_script,')
275                 print('and_upload_the_code_to_your_
                Arduino.')
276                 print('Exiting...')
277
278         print("\nTaking_data...")
279         print("Press_ctl+c_to_terminate_process")
280
281         if nDetectors>1:
282             for i in range(nDetectors):

```

```

283         string_of_names += detector_name_list[i]
284         +',␣'
285
286     else:
287         string_of_names+=detector_name_list[0]
288
289     #print(string_of_names)
290     file.write('Device␣ID(s):␣'+string_of_names)
291     file.write('\n')
292     #detector_name = ComPort.readline()      # Wait and read
293     data
294     #file.write("Device ID: "+str(detector_name))
295
296     while True:
297         for i in range(nDetectors):
298             if globals()['Det%s' % str(i)].inWaiting
299             ():
300
301                 #data = globals()['Det%s' % str(i)
302                 ].readline().replace(
303                 # '\r\n','')      # Wait and read
304                 data
305
306                 data = globals()['Det%s' % str(i)
307                 ].readline().decode(
308                 ).strip()      # Wait
309                 and read data
310
311                 file.write(str(datetime.now())+"␣
312                 "+data+"␣"+detector_name_list[
313                 i]+'␣\n')
314                 globals()['Det%s' % str(i)].write
315                 (str.encode('got-it'))
316
317         for i in range(nDetectors):
318             globals()['Det%s' % str(i)].close()
319         file.close()
320

```

```

307 if mode == 2:
308
309     cwd = os.getcwd()
310     dir_path = input("\nEnter location to save SD data (
311                     default: " + cwd + "/SDFiles):")
312     if dir_path == '':
313         dir_path = cwd + "/SDFiles"
314
315     if not os.path.exists(dir_path):
316         os.makedirs(dir_path)
317
318     signal.signal(signal.SIGINT, signal_handler)
319     ComPort = serial.Serial(port_name_list[0]) # open the COM
320     Port
321     ComPort.baudrate = 9600 # set Baud rate
322     ComPort.bytesize = 8 # Number of data bits
323     = 8
324     ComPort.parity = 'N' # No parity
325     ComPort.stopbits = 1
326
327     if ComPort.readline().strip() == 'CosmicWatchDetector':
328         time.sleep(1)
329         detector_name = ComPort.readline()
330
331         print('\n--Detector Name--')
332         print(detector_name)
333
334         ComPort.write("read")
335         counter = 0
336
337         while True:

```

```

337         data = ComPort.readline()      # Wait and
                                         read data
338         #print(data)
339         if 'Done' in data:
340             ComPort.close()
341             sys.exit()
342         elif 'opening:' in data:
343             fname = dir_path + '/' + data.split('
')[-1].split('.txt')[0]+'.txt'

344
345             print("Saving to:" + fname)
346             file = open(fname, "w",0)
347             counter = 0
348
349
350             elif 'EOF' in data:
351                 file.close()
352
353             else:
354                 file.write(data)
355                 counter +=1
356
357     else:
358         print('---Error---')
359         print('You are trying to read from the SD card.')
360         print('Have you uploaded SDCard.ino to the
        Arduino?')
361         print('Is there a microSD card inserted into the
        detector?')
362         print('Exiting...')
363         sys.exit()
364
365 if mode == 3:

```

```

366     print("\nAre you sure that you want to remove all files from
        SD card?")
367     ans = input("Type y or n:")
368     if ans == 'y' or ans == 'yes' or ans == 'Y' or ans == 'YES':
369         signal.signal(signal.SIGINT, signal_handler)
370         ComPort = serial.Serial(port_name_list[0]) # open the COM
            Port
371         ComPort.baudrate = 9600                # set Baud rate
372         ComPort.bytesize = 8                    # Number of data bits =
            8
373         ComPort.parity = 'N'                    # No parity
374         ComPort.stopbits = 1
375
376     if ComPort.readline().strip() == 'CosmicWatchDetector':
377         time.sleep(1)
378         ComPort.write("remove")
379         while True:
380             data = ComPort.readline()          # Wait and read data
381             print(data)
382             if data == 'Done...\r\n':
383                 print("Finished deleting files.")
384                 break
385         ComPort.close()
386         sys.exit()
387
388     else:
389         print('---Error---')
390         print('You are trying to remove files from the
            microSD card.')
391         print('Is there an SD card inserted?')
392         print('Do you have the correct code on the Arduino?
            SDCard.ino')
393         print('Exiting...')
394         sys.exit()

```

```

395
396     else:
397         print("Exiting...")
398         sys.exit()
399
400 if mode == 4:
401     bg = DataCollectionProcess(queue)
402     #bg.daemon = True
403     #bg.start()
404     thread.start_new_thread(RUN, (bg,))
405     #p=multiprocessing.Process(target=RUN)
406     #p.start()
407     #server stuff
408     application = tornado.web.Application(
409         handlers=[(r'/', WSHandler)]
410     )
411     http_server = tornado.httpserver.HTTPServer(application)
412     port = 9090
413     http_server.listen(port)
414     myIP = socket.gethostbyname(socket.gethostname())
415     print('CosmicWatch_detector_server_started_at%s:%d' % (myIP,
416         port))
417     print('You can now connect to your device using http://
418         cosmicwatch.lns.mit.edu/')
419     mainLoop = tornado.ioloop.IOLoop.instance()
420     #in the main loop fire queue check each 100ms
421     try:
422         scheduler = tornado.ioloop.PeriodicCallback(checkQueue,
423             100, io_loop = mainLoop)
424     except:
425         # io_loop arguement was removed in version 5.x of Tornado
426         .
427         scheduler = tornado.ioloop.PeriodicCallback(checkQueue,
428             100)

```

```
424     scheduler.start()  
425     #start the loop  
426     mainLoop.start()
```