E90: Building and Calibrating a Geiger Meter and Mapping Radiation

Will Gilchrist and Rachel Wesley
Advisor: Professor Lynne Molter
Spring 2019
Introduction

Radiation commonly refers to the photons or subatomic particles that are expelled from atomic nuclei during decay or fission. It is an unavoidable fact of daily life; the body is bombarded with radiation every minute. Sources include heavy elements in the air and ground as well as cosmic gamma rays. Small amounts of radiation are effectively harmless. However, long term exposure to excess radiation increases the probability of cancerous mutations, as the high energy events have some small chance to damage crucial segments of DNA.

Unaided, a human cannot detect radiation. A wide range of devices can detect radiation with varying levels of precision depending on the type of radiation. In this project, we construct and calibrate two Geiger-Muller meters, or Geiger meters for short. While simple, these detectors are effective at detecting a wide range of radiation. We also calibrate the devices with the end goal of geographically measuring the radiation of Swarthmore College.

This report details the construction and calibration of the Geiger meters as well as the data collection process. We present 3D heatmaps of radiation around Swarthmore College.

Background

A Geiger-Muller meter is a common radiation detection device. Most people know them from movies as the rod-shaped devices that produce clicking noises when waved over radioactive material. Geiger-Muller meters are exceptionally good at detecting most radiation. The rod portion contains a select mixture of relatively inactive gases that ionize when struck with a gamma ray or charged particle. The ionized gas molecule travels through the mixture towards a cathode, ionizing other molecules as it travels. The result is significant impulse in voltage, which produce the trademark “click” when attached to a speaker device. The traveling cascade of electrons in the tube is referred to as a Townsend avalanche. The circuitry and physics involved are relatively simple; the tube itself is effectively a large capacitor, and the rest of the circuitry deals with analog to digital conversion and filtering. The challenge lies in correctly, and safely, constructing and calibrating the device.

The standard design for a Geiger Muller meter is a cylindrical tube with a wire through the center. The tube is filled with a single gas, such as argon, that is ionized by beta particles, alpha particles, or gamma rays, depending on the gas. The wire at the center of the tube is held at a high voltage (400V-600V) and the shell is grounded. When the argon is ionized, the free electrons make their way to the center wire. Along the way, they ionize more gas molecules, causing the aforementioned electron avalanches. As the electrons reach the anode, they generate a measurable current. The electron avalanche amplifies a single radiation event to the degree that little to no amplification of the electrical signal is necessary. Figure 1 below shows a basic diagram.
Figure 1: Basic GM Counter (Knoll 217)

The tube and resistor are effectively an RC circuit, and the output signal rises and falls exponentially as one would see in a rapidly charged and discharged capacitor.

Geiger meters have some drawbacks as detection devices. While they are a cheap and effective means of detecting radiation, they are poor at distinguishing between different radiation types. The strength and type of a radiation event is obscured by the electron avalanche; the same strength signal comes through for an alpha particle as for a gamma ray, two fundamentally different types of radiation. Choice of ionizable gas and tube materials can exclude certain types of radiation. For example, one of our tubes is capable of detecting alpha particles, beta particles, and gamma rays, and the other can only detect beta particles and gamma rays.

Geiger meters also have a refractory period in between radiation events. As charged particles “avalanche” through the tube and stick to the anode and cathode, the tube is “frozen” and any radiation events during that time period do not trigger Townsend avalanches. The refractory period, or dead time, is proportional to the time constant of the RC circuit as it appears in Figure 1. This dead time is negligible for background radiation doses, though we did witness it while calibrating with actively decaying radiation sources. The tubes also have a tendency to lose effectiveness over time, but this is negligible over the scale of our project.

Constructing the Geiger Meters
Design

We constructed two Geiger Meters, both capable of detecting beta and gamma radiation, and one capable of detecting alpha radiation. Geiger Meters utilize high DC voltages to allow noble gasses to react with radiation; in the interest of safety, we built two Geiger Meters designed by Images Scientific Instruments, who provided the silk boards and tubes. The designs are identical except for the gas tubes and load resistors. A schematic for the GCK-02 we used is shown in Figure 2.
A direct or alternating current source is stepped up to supply between 400V and 600V across the gas tube. Following a radiation event, the tube outputs a signal as shown in Figure 3, which is the same output for a charging and discharging capacitor.
The tube output is fed through a comparator, giving a square wave for digital analysis seen in Figure 4. Additional comparators clean up the signal and then feed it to output port as well as a speaker and LED loop for qualitative detection.

One design used a GMT-01 Geiger Counter tube. The tube contains neon and halogen. It detects gamma and beta radiation through all sides and alpha radiation through a mica window on the front of the tube. The tube has an optimal operating voltage of 500V. The circuit for this Geiger meter differed by having an anode resistor of 10MΩ and jumper that shorts out diode D8 and provides 400V DC to the GMT-01 tube. The tube has a dead time of 90ms.

The second design was identical to the first, with a few distinctions. The second design used the GMT-02 glass Geiger Counter tube. The neon-halogen tube is capable of detecting beta and gamma radiation. The anode resistor is 2.2MΩ and the dead time is 80µs.

**Soldering: Some Early Difficulties**

Each Geiger Meter tube requires a different anode and cathode resistor in their circuit design. When assembling the circuit for GMT-02, there were discrepancies between the instruction manual delivered in the kit and the instructions on the website for the specific value of the anode (R4) resistor. This lead to us soldering in the wrong resistor. When removing the resistor, we damaged one of the conducting ring on half the board for both sides of the resistor connection. Luckily, the portion of the conducting ring that was removed was not connected to the conducting side of the silk board. This means the mistake did not impact the performance of this Geiger Meter, but is important to mention. This mistake was not repeated at any point and we had no issues soldering the other components.
Safety

It is important to take proper safety precautions when dealing with high voltage and current. There are a total of 8 capacitors on each of the circuit boards. Capacitors can potentially hold a charge for a long time. We were dealing with circuits that utilize 400V and 500V of potential. This voltage is potentially deadly if not handled correctly considering the transformer supplies a large current (1A). For reference, static electricity is many orders of magnitude higher in voltage, but there is no current, thus it is not dangerous. We wanted to take precautions should we accidentally ground ourselves while touching a high voltage component of the circuit. Accidental discharge of a capacitor could lead to minor burns or potentially death depending on the current path and difference in potential. Whenever handling either circuit, we made sure to wear shoes with rubber soles in order to prevent that connection to ground. We were also mindful of how we handled the circuits and any loops we could potentially close when touching the boards.

The lethal amount of current is 0.1A. Our wall-adapter outputs 1A of current, which is 10 times higher than the lethal amount. We deduced through Ohm’s Law, V=I/R, that the minimum resistance of gloves needed in order to stop a lethal level of current (.1A) at 500V is 5000 ohms. We obtained a set of medical gloves and tested the resistance of them with an ohm meter. The ohm meter was unable to give us an exact resistance reading of the gloves as they were too high in resistance. They are well above the minimum resistance needed to prevent us from completing a circuit should we touch a charged capacitor lead.

Proper storage of the circuits is another point of concern. Since the circuit boards are soldered together, there are exposed leads coming on the bottom of the board. This could be a fire hazard, should the board rest on a conductive, flammable surface. To solve this we secured each of boards to a plastic box with stands on the edges and corners. These stands were screwed into the plastic box to lock and hold the board in place. The stands elevate the circuit boards off the back of the box and make it hard for one to accidentally touch the exposed solder. See Figure 5 below for the case and stand design.

Figure 5: Case and Stand Design with GMT-01 Pictured
In order to prevent any other accident we made sure to always work on the circuits together and/or under direct supervision of Professor Molter. We made sure the workspace was clean and neat before completing any construction or testing. We also removed any food or beverage around the work station.

**Sampling**

In order to measure radiation readings from the geiger meters, we installed an Arduino Uno to log when an event occurred in the serial monitor. The Arduino Uno samples at a rate of 496 samples/second with its ADC. We designed the code to increment a counter on the serial monitor when a radiation event was detected. There is also a timer programmed to print how many minutes have passed since running the individual experiment along with how many counts were recorded at that minute. This allowed us to find the radiation counts per minute through simple subtraction. For ease of convenience, when taking our actual data, we adapted the code to output the counts per minute, which can be seen in Appendix B.

Initially, we found that the pulse width of a radiation event was too narrow for polling. This meant that the Arduino Uno would miss a significant amount of radiation events when sampling. Figure 6 below depicts the issue we faced.

![Figure 6: Initial Sampling Issue](image)

In Figure 6 it is clear that if a radiation event occurred between when the Arduino Uno was sampling, the event would be missed. This created significant error in the early data. To remedy this, we created an interrupt service routine (ISR) so that the Arduino Uno records a radiation event whenever a rising edge was detected. This code can be found in Appendix A. Additionally, we adjusted the potentiometer to increase the measured pulse width to improve the readings of the Arduino. With these two adjustments, we were able to log most background events that triggered the Geiger meter. Figure 7 shows the adjusted pulse width with both the analog and digital outputs from the Geiger meter.
One disadvantage to the ISR is that sometimes, seemingly randomly, the data logger records there were two radiation events for only one audible click of the Geiger meters. We refer to this as being “double count” error. This error could be from a number of things. There is a possibility the ISR Arduino code is not functioning properly. It is also possible that the Arduino Uno is extremely sensitive to external movement and thus needs to be completely immobile when taking samples. We found that a simple tap of the Arduino Uno would cause this double count phenomena to occur. By securing a plastic casing to the back of the Arduino we were able to reduce the double count behavior, but not entirely. There is also the possibility that there are two radiation events so close together such that the speaker makes a single tone, and the data logger is actually correct. Some elements do release consecutive gamma rays while decaying. Whatever the cause, this error, if it is one, is systematic. During the calibration process, we sought to account for this error and used measurement techniques to minimize its occurrence.

**Calibration**

Geiger meters are excellent for qualitative radiation detection in that they give no information concerning the energy or type of radiation. However, they do provide a dependable method for measuring the volume of radiation, i.e., the number of gamma rays and beta particles, collectively referred to as events, per unit of time. For each radiation event, there is some probability $\alpha$ that the event makes contact with a gas molecule and starts the detectable electron avalanche. For a tube with known $\alpha$, it is possible to extrapolate measurements for an accurate estimate of the true radiation.
For calibration, we used a 0.91µCi Cs-137 radiation source. The decay scheme of Cs-137 is shown in Figure 8.

![Cs-137 Decay Scheme](image)

Figure 8: Cs-137 Decay Scheme

Approximately 44.6% and 55.4% of the events output from Cs-137 are gamma and beta radiation, respectively. The energy levels of Cs-137 events are all high enough energy to trigger electron avalanches in both tubes. Theoretically, the value of $\alpha$ ought to vary for each type of radiation. Determining each $\alpha$ would require any calibration to involve multiple radiation sources with differing ratios of beta and gamma output. However, our goal is to measure background radiation. If the background radiation sufficiently matches the calibration source radiation in composition, we may use a single $\alpha$ value. According to the EPA, about 91% of background radiation in Pennsylvania is due to atmospheric Radon decay\(^2\). Radon primarily decays into alpha and beta particles. A majority of the Radon decay output is alpha particles. While alpha particles are the most molecularly destructive, they are also the most easily blocked. We considered the atmospheric alpha decay to be negligible. Considering the ratio of beta decay from Radon in addition to cosmic and terrestrial gamma ray sources, we concluded Cs-137 sufficiently matches the background radiation. Thus, we accept an $\alpha$ value from Cs-137 to be sufficiently accurate.

To determine the $\alpha$ value of a detector, we need to know the actual volume of radiation passing through the detector in counts per minute. The Cs-137 source had an initial activity of 1µCi. Adjusting for decay since manufacture date, the source at time of writing has an activity of 0.91µCi. The source, which is modeled as a point, radiates in all directions. The radiation flux at some distance from the source varies by an inverse square relation. The relationship is

\( N = \frac{137\, \text{meV}}{r^2} \)

\(^1\) Nuclear Fission Fragments. [http://hyperphysics.phy-astr.gsu.edu/hbase/NucEne/fisfrag.html](http://hyperphysics.phy-astr.gsu.edu/hbase/NucEne/fisfrag.html).

\(^2\) Calculate Your Radiation Dose | Radiation Protection | US EPA.
derived via Gauss’s law, and can be geometrically intuited as the total radiation divided by the surface area of a sphere with radius to some point. The radiative flux of a source is given by

$$\Phi = \frac{P}{4\pi (r + \delta r) \sqrt{4\pi}}$$

where $P$ is the activity of the source and $r$ is the distance from the source. A factor of $\frac{1}{\sqrt{4\pi}}$ is included to adjust for small $r$. Multiplying the flux by the cross sectional area of the Geiger tube gives a theoretical value for events per unit time. This is true for an isolated source. In practice, we have to correct for background radiation as well as gamma ray reflections. In initial data collection, gamma rays from the source reflected off some metal surface and interfered with incident gamma rays; we witnessed an unusual dip in efficiency at one distance from the source, indicative of destructive interference.

Our general strategy for calibration was as follows: record the events per minute for each detector at various distances from the source, calculate the theoretical events per minute, correct for background, and divide the measured by the theoretical for an $\alpha$ value. As mentioned, we witnessed some unusual dips in efficiency in the initial dataset. Gamma rays, as electromagnetic radiation, can reflect off metal. Reflected gamma rays can destructively interfere with incident gamma rays, causing unusual dips in efficiency. In subsequent measurements, we surrounded the detector and source with lead shielding to prevent reflection and interference. Figure 9 shows our initial measurement setup compared to subsequent setups with lead shielding.
Note that the shielded setup also involved a lead roof (not shown).

Radiation is a completely random process. The average event rate of the source is well documented, but the decay process on a macroscopic scale is completely probabilistic. The decay of radioactive material is well modeled as a Poisson distribution. Macroscopically, this is of little concern as we are primarily focused on average values. However, this has important implications when we consider the dead time in a Geiger Muller tube. All tubes have a deadtime (DT) that occurs after a successful event detection. During the time, the device is unable to detect another event, as the electrons that previously avalanched towards the anode must diffuse back into the gas. This places limits on how much radiation can be detected, as well as how likely we are to detect an event. Using a Poisson distribution based model, we can adjust our theoretical values to account for the DT of a tube.

Consider a length of time \( T_{DT} \), the length of a time of a tube's deadtime. Say that \( k \) events pass through the tube during that time, and there is some probability \( \alpha \) that an event ionizes a gas molecule. Figure 10 illustrates the process.

![Figure 10: Dead Time](image)

Within this time period, as each event arrives, it can be detected or not detected. Detection leads to a deadtime and failure to detect the remaining events. The probability of completely missing all \( k \) events is \( (1 - \alpha)^k \). Otherwise, at most one event will be detected during this time period. Modeling the radiation source as a Poisson distribution allows us to account for the possibility of any \( k \) value. For some time \( T_{DT} \) the probabilities of detection are

\[
P_0 = \sum_{k=0}^{\infty} \frac{k^k \lambda^k}{k!} (1 - \alpha)^k
\]

\[
P_1 = 1 - P_0
\]
where $\lambda$ is the flux adjusted for the time $T_{DT}$. Multiplying $P_1$ by $60/T_{DT}$ gives the expected value per minute accounting for the deadtime. The series collapses nicely giving final expression for counts per minute is given by

$$CPM = \frac{60}{T_{DT}} (1 - e^{-\lambda \cdot (\alpha - 1)})$$

CPM in this case is the empirically measured value with known error. With some algebraic manipulation, we can solve for the efficiency

$$\alpha = -\frac{1}{\lambda} \ln(1 - CPM \left(\frac{60}{T_{DT}}\right))$$

The probability of an event arriving during a deadtime for background radiation is negligibly small. Once $\alpha$ is known, we need only divide future background measurement to get an estimate of the actual radiation present.

We took several large measurements at varying distances from the source, and attempted to explicitly solve for $\alpha$ as detailed above. Unfortunately, the resulting $\alpha$ values were sporadic, with one measurement yielding a physically unrealizable $\alpha$ value. The most unrealistic $\alpha$ values were calculated at close distances. We suspect there were some other sources of error that were exacerbated by the rapid arrival rate. These include ISR build up, when multiple interrupts occur before the current one is executed, and ISR double counts, which seemed to systematically occur for one third of background events.

To account for deadtime and other issues, we took final calibration measurements at a distance that minimized the effect of deadtime. At 15 cm, we considered the deadtime issue to be negligible. The probability of missing one or more due to deadtime, in some time frame equal to a dead time, is given by:

$$P_{miss} \geq 1 = \sum_{k=2}^{\infty} \frac{k^k e^{-\lambda}}{k!}$$

where $\lambda$ is the average arrival rate, or total flux, for this time period. Running the code in Appendix E gives the flux value for the GMT-02 tube which we used for the mapping process due to its short deadtime and equidirectional detection. Over the course of a minute, our desired sample time, we found the probability of missing one or more radiation event due to deadtime to be 0.4530. On its face, this value is not negligible. However, if we evaluate the poisson distribution at $k = 2$, we find the probability of missing just one event in a minute to be 0.4459 leaving a 0.0071 probability of missing more than 1 event in a minute. The probability of missing a significant number of events is unlikely. Even though the probability of missing one
event is somewhat likely, a single event is negligible relative to the random variations in radiation.

We measured radiation for 40 minutes 15 cm away from the source and for 40 minutes in the same location without a source to account for background. Each minute counted as a sample. Dividing the measured radiation by the theoretical flux at that distance, we found an $\alpha$ value with 95% confidence interval of

$$\alpha = 0.0319 \pm 0.0562$$

The large error is due to propagation of the variation in background data and the source. The lower bound ends at $\alpha = 0$, as a negative value is not physically realizable. Running the code in Appendix E calculates our $\alpha$ value. Dividing measured counts per minute by $\alpha$ should give an estimate of the actual radiation present.

This efficiency is comparable to other Geiger meters which generally detect gamma rays with about 3% efficiency depending on build and gas mixture. However, our source emitted 55.4% beta particles which should be detected with close to 100% efficiency.\(^3\) We suspect this may be due to the glass walls of our tube. Geiger meters with metal gas containers are exceptionally good at detecting charged particles, as the particles need not even make it into the gas. They can ionize the metal casing, and the electron finds it way into the gas, triggering the electron cascade. Our primary tube has glass walls which are not conductive, so if a beta particle ionized some glass molecule, the free electron is unlikely to make its way into the gas. This could be the cause of our lower than expected efficiency.

**Mapping Swarthmore**

**GPS Chip Integration**

In order to map an area, we needed to incorporate locational data. We decided to add a GPS chip that was compatible with the Arduino Uno. The GPS chip we used was the NEO-6M GPS Module, pictured below in Figure 11.

---

\(^3\) Knoll, Glenn F. Radiation Detection and Measurement.
We pulled just the latitude and longitude data from this chip. We outputted this information for each counts per minute statement to the serial monitor.

Advantages to using this chip is that it was easy to pull the data components we were interested in, which were just latitude and longitude. This chip also offers other data like time, altitude, and which satellites were being accessed. A disadvantage to this chip is that it does not work inside buildings. This was not an issue for the scope of our project and could be fixed by adding an external antenna.

Collecting Data

After successful calibration and GPS chip integration, the next step was to actually collect data of Swarthmore College’s campus. In order to do this, we replaced the 1A wall outlet power supply with a 9V battery. The circuit uses a zener diode configuration to step up a low voltage while sacrificing some current. For collecting data, we used the GMT-02 device because of its non directional properties. The full setup in a mobile cart can be seen in Figure 12 below.
The path we took began and ended at Hicks Engineering Building, marked with a red circle on the map in Figure 13. We went towards Science Center, Parrish, Worth, Alice Paul/David Kemp, Sharples, Dana/Hallowell, Wharton, and Whittier. There was a mixture of walking around academic buildings, dorms, and open space. We walked around continuously and the Arduino reported a count of radiation in a minute and the longitude and latitude at the location it logged the radiation count. Figure 13 shows the route traveled.
Ideally, if someone wanted to collect data to create a full continuous map, a person would move a certain distance, wait for a minute until the Arduino outputted the data to the serial monitor, and then move to the next location to repeat this process. This method would be more meticulous and require more time. We collected readings for discrete points along the path in Figure 13, but not the full continuous path we walked.

In order to take data with the least variations possible, one would have to take data at night. Taking data during night eliminates the variable of any radiation from the sun. Unfortunately, the week leading up to our presentation, and this report, there was forecasted rain every single night. This stopped us from taking the ideal data. The second best option, which we resorted to, was to take data on a cloudy day.

Creating the Map

In order to analyze the data we collected, we utilized MATLAB to create the desired heatmaps. We assume there is always background radiation present. This background level was calculated through taking the average of all of measured radiation data. We then created Gaussian curves to account for confidence in the location of our collected data. The heights of the Gaussian curves are calculated by dividing the difference between the measured radiation at some point and the average radiation by the efficiency of our device, or $\alpha$. The standard deviation of the Gaussian curves are derived from sampling error in the GPS. All GPS devices are scrambled by the US military to have an accuracy of $\pm 10$ m. This equates to approximately 0.0001 degrees of longitude or latitude in Pennsylvania. These calculations can be found in Appendix I. These Gaussian curves are then added to the background radiation to produce an interactive 3D heatmap in MATLAB. Screenshots of various views of the heatmap can be seen in Figures 14-16.
Figure 14: 2D Heatmap View (Longitude vs Radiation Count)

Figure 15: 3D Heatmap View with Gaussian Curves and Swarthmore’s Campus Underneath
By adding the Gaussian curves, we are able to have the location where the measurement was taken have a heavier impact on the heatmap. When moving away from that measurement, the added value will have less of a weight on the heatmap, in a decreasing fashion like a Gaussian curve. In Figure T and Figure U, the locations with higher levels of radiation are yellow-orange curves above the average plane and locations with lower levels of radiation are green-blue curves below the average plane. There are portions of the heatmap with no peaks at all. These peaks are places no data was taken and so we assume there is the average background radiation present. Figure V shows the actual corresponding locations to the data. The yellow-orange spots are places measured with having higher radiation, the green-blue spots are places measured with having lower radiation, and the green spots are places unmeasured where we assume there is average background radiation.
Results

Through our data collection, we did not find any statistically significant variations in radiation. We found the maximum radiation measurement to be 1066 counts per minute, the minimum to be 336 counts per minute, and the average to be 742 counts per minute from the heatmap. The EPA publishes radiation data on their website, which can be found in Figure 17 below.

Figure 17: EPA Radiation Data for Philadelphia, PA

The EPA published data is on a log scale. The EPA found around a little less than 3000 counts per minute of radiation in Philadelphia, PA. If we roughly compare our measurements with the EPA's, we believe our measurements are acceptable. We are not able to directly compare our measurements, as we did not necessarily use the same device as the EPA. If we roughly compare our measurements with the EPA's, we believe our measurements are acceptable.

Geiger meters are a good way of qualitatively measuring radiation. They are good at finding if an area generally has higher or lower levels of radiation, not exactly what the radiation levels are. In order to get exact measurements, extremely sophisticated and more expensive equipment is necessary. The EPA found around 1.5 to 2.5 counts per minute of radiation, not exactly what the radiation levels are. We assume the EPA has a more efficient detector.

Geiger meters are a good way of qualitatively measuring radiation. They are good at finding if an area generally has higher or lower levels of radiation, not exactly what the radiation levels are. In order to get exact measurements, extremely sophisticated and more expensive equipment is necessary. The EPA found around 1.5 to 2.5 counts per minute of radiation, not exactly what the radiation levels are. We assume the EPA has a more efficient detector.

Figure 17: EPA Radiation Data for Philadelphia, PA

Below, the EPA publishes radiation data on their website, which can be found in Figure 17. We found the maximum radiation measurement to be 1066 counts per minute, the minimum to be 336 counts per minute, and the average to be 742 counts per minute from the heatmap. We found the maximum radiation measurement to be 1066 counts per minute, the minimum to be 336 counts per minute, and the average to be 742 counts per minute from the heatmap. We found the maximum radiation measurement to be 1066 counts per minute, the minimum to be 336 counts per minute, and the average to be 742 counts per minute from the heatmap. We found the maximum radiation measurement to be 1066 counts per minute, the minimum to be 336 counts per minute, and the average to be 742 counts per minute from the heatmap. We found the maximum radiation measurement to be 1066 counts per minute, the minimum to be 336 counts per minute, and the average to be 742 counts per minute from the heatmap. Based on our data collection, we did not find any statistically significant variations in radiation.

Results
Conclusion

We have successfully constructed two different Geiger Muller meters. The meter with the GMT-01 tube is directional, as it measures radiation at a significantly higher rate through its front mica window. The meter with the GMT-02 is not directional and measures radiation in all directions through a glass surface. We decided to use the GMT-02 for mapping campus, as background radiation comes from any direction, and the directional nature of the GMT-01 threatened to skew geographic data. In calibration, we found the GMT-02 to be approximately 3% efficient. We then integrated the GMT-02 with an Arduino Uno and GPS chip. The GPS chip outputted the latitude and longitude of our sampling locations. This allowed us to successfully create a 3D heatmap of Swarthmore College’s radiation levels. Our findings are not unreasonable relative to data provided by the EPA for radiation levels in Philadelphia, PA. We did not find any statistically significant abnormalities in radiation around Swarthmore College. A large confidence interval on from the calibration process trivialized any variations we detected. Though it does not corroborate our data, the data provided by the EPA does not conflict with our data. The EPA generally recorded higher radiation levels, which is expected as the organization likely has larger and more sophisticated means of detection.
Acknowledgements

1) Professor Lynne Molter, Engineering Department  
2) Professor Hillary Smith, Physics Department  
3) Ed Jaoudi, Engineering Department  
4) Cassy Burnett, Engineering Department  
5) Professor Allen Moser, Engineering Department  
6) Professor Michael Piovoso, Engineering Department

References

Calculate Your Radiation Dose | Radiation Protection | US EPA.


How to Interface GPS Module (NEO-6m) with Arduino - Arduino Project Hub.


Appendices

Appendix A: Arduino: Interrupt Service Routine Code Used in Calibration (No GPS)

```c
const int GMinput = 2; // input

// variables will change:
volatile int eventsDetected = 0; // variable for reading the pushbutton status
int minutesPassed = 0;
int oldMinutes=0;

void setup() {
  Serial.begin(9600);
  pinMode(GMinput, INPUT);
  // Attach an interrupt to the ISR vector
  attachInterrupt(0, pin_ISR, RISING);
  delay(1000);
}

void loop() {
  minutesPassed =millis()/60000.0;
  if (minutesPassed-oldMinutes>0){
    oldMinutes=minutesPassed;

    Serial.print(eventsDetected);
    Serial.print("\n");
    eventsDetected = 0;
  }
}

void pin_ISR() {
  eventsDetected++;
  //Serial.print(eventsDetected);
}
```
Appendix B: Arduino: Interrupt Service Routine Code Used in Mapping (GPS)

#include <SoftwareSerial.h>
#include <TinyGPS++.h>

// The TinyGPS++ object
TinyGPSPlus gps;

const int GMinput = 2;     // input
// The serial connection to the GPS module
SoftwareSerial ss(4, 3);
int GMPin= 7;
int val=0;
static const uint32_t GPSBaud = 9600;

volatile int eventsDetected = 0;
int minutesPassed = 0;
int oldMinutes=0;

void setup() {
    Serial.begin(9600);
    pinMode(GMinput, INPUT);
    pinMode(GMPin, INPUT);
    // Attach an interrupt to the ISR vector
    attachInterrupt(0, pin_ISR, RISING);
    delay(1000);
    ss.begin(GPSBaud);
}

void loop() {

    while (ss.available() > 0){
        minutesPassed =millis()/60000.0;
        gps.encode(ss.read());

        if (minutesPassed-oldMinutes>0){
            oldMinutes=minutesPassed;
            Serial.print(minutesPassed);
            Serial.print("n");
            Serial.print(eventsDetected);
            Serial.print("t");
        }
    }
}
// Serial.print("Latitude= ");
Serial.print(gps.location.lat(), 6);
Serial.print("t");
//Serial.print(" Longitude= ");
Serial.println(gps.location.lng(), 6);
Serial.print("n");
eventsDetected=0;
}
}
}

void pin_ISR() {
    eventsDetected++;
    //Serial.print(eventsDetected);
    // Serial.print("n");
}

Appendix C: MATLAB: Trim Rows Function
Filename: trimRows.m

function out=trimRows(rowMat)
    %trims NaN of column vector
    out=rowMat;
    for i=1:length(rowMat)
        if isnan(rowMat(i))
            out(i:end)=[];
            break
        end
    end
end
Appendix D: MATLAB: Calibration GMT-02 Data Excel Sheet
Filename: GMT02LongerData.xlsx
Sheetname: Sheet1

<table>
<thead>
<tr>
<th>Background</th>
<th>15cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>39</td>
</tr>
<tr>
<td>27</td>
<td>39</td>
</tr>
<tr>
<td>11</td>
<td>59</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
</tr>
<tr>
<td>29</td>
<td>46</td>
</tr>
<tr>
<td>17</td>
<td>43</td>
</tr>
<tr>
<td>15</td>
<td>53</td>
</tr>
<tr>
<td>12</td>
<td>34</td>
</tr>
<tr>
<td>22</td>
<td>41</td>
</tr>
<tr>
<td>14</td>
<td>46</td>
</tr>
<tr>
<td>28</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>51</td>
</tr>
<tr>
<td>16</td>
<td>27</td>
</tr>
<tr>
<td>22</td>
<td>38</td>
</tr>
<tr>
<td>24</td>
<td>51</td>
</tr>
<tr>
<td>13</td>
<td>35</td>
</tr>
<tr>
<td>25</td>
<td>44</td>
</tr>
<tr>
<td>14</td>
<td>37</td>
</tr>
<tr>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>17</td>
<td>50</td>
</tr>
<tr>
<td>18</td>
<td>33</td>
</tr>
<tr>
<td>18</td>
<td>39</td>
</tr>
</tbody>
</table>
Appendix E: MATLAB: Calibration

Note: Appendix C-E must be saved in same MATLAB folder in order to run this code

%% Import Data

% Import into arrays
GMT02_Raw_Table = readtable('GMT02LongerData.xlsx','Sheet','Sheet1','Range','A1:B42');
GMT02_Raw = table2array(GMT02_Raw_Table);

% Pull columns
GMT02_BG=GMT02_Raw(:,1);
GMT02_15cm=GMT02_Raw(:,2);

% Trim column
GMT02_BG=trimRows(GMT02_BG);

%% Theoretical Values

r=[inf .15]; %meters
P=0.9103e-6; %Curies
countsPerCi = 3.7e10; %counts/s/Ci

% flux/m^2
flux = P*countsPerCi./(4.*pi.*((r+(1./sqrt(4.*pi))).^2));
% counts/s/m^2

%areas
GMT02_Area = 0.0095*0.1;

% theoretical flux through each tube
GMT02_flux = 60*flux*GMT02_Area; %counts/m

% figure(1)
% plot(r,GMT02_flux)
% title('Theoretical Fluxes NO Poisson Correction')
% xlabel('r')
% ylabel('CPM')
% legend('GMT02')
%% Analysis

% find averages for each distance
Detected_Avg_GMT_02=[mean(GMT02_BG), mean(GMT02_15cm)];

% find standard deviation for each distance
Detected_STD_GMT_02=[std(GMT02_BG), std(GMT02_15cm)];

% 95% confidence interval
error_GMT02=[Detected_Avg_GMT_02-2.*Detected_STD_GMT_02;Detected_Avg_GMT_02 +2.*Detected_STD_GMT_02];

%% Theoretical Values with Poisson Correction

% deadtime
GMT02_DT = 80e-6; % s/DT

% flux per deadtime
GMT02_flux_perDT = GMT02_flux*(1/60)*GMT02_DT; % lambda 02

%% Real Data Analysis

Error_Correction=sqrt(2*(Detected_STD_GMT_02(2)).^2+2*(Detected_STD_GMT_02(1)).^2);
Difference_Correction=(Detected_Avg_GMT_02(2)-Detected_Avg_GMT_02(1));

GMT02_Corrected_15cm=[Difference_Correction + Error_Correction; Difference_Correction-
>Error_Correction]; % 15cm-BG

% calculate alpha
alpha=Difference_Correction./GMT02_flux(2);
Error_alpha=abs(alpha).*sqrt((Difference_Correction./Error_Correction).^2+(2*sqrt(GMT02_flux(2))./GMT02_flux(2)).^2);

alpha_95_confidence=[alpha+ Error_alpha;alpha - Error_alpha]

% nbins=15;
% subplot(2,1,1)
% histogram(GMT02_15cm,nbins)
%
% subplot(2,1,2)
% histogram(GMT02_BG,nbins)
Appendix F: MATLAB: Data Collection Excel Sheet
Filename: Data Collection.xlsx
Sheetname: May 1
Column 1: Counts per minute (of Radiation)
Column 2: Latitude
Column 3: Longitude

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>39.906639</td>
<td>-75.354927</td>
</tr>
<tr>
<td>23</td>
<td>39.906673</td>
<td>-75.3554</td>
</tr>
<tr>
<td>29</td>
<td>39.906291</td>
<td>-75.355712</td>
</tr>
<tr>
<td>30</td>
<td>39.905838</td>
<td>-75.355308</td>
</tr>
<tr>
<td>30</td>
<td>39.905384</td>
<td>-75.354957</td>
</tr>
<tr>
<td>15</td>
<td>39.90496</td>
<td>-75.354774</td>
</tr>
<tr>
<td>23</td>
<td>39.904949</td>
<td>-75.354293</td>
</tr>
<tr>
<td>32</td>
<td>39.905242</td>
<td>-75.353736</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>------------------</td>
</tr>
<tr>
<td>29</td>
<td>39.905601</td>
<td>-75.353675</td>
</tr>
<tr>
<td>17</td>
<td>39.905761</td>
<td>-75.354026</td>
</tr>
<tr>
<td>27</td>
<td>39.906154</td>
<td>-75.354133</td>
</tr>
<tr>
<td>29</td>
<td>39.906356</td>
<td>-75.353584</td>
</tr>
<tr>
<td>19</td>
<td>39.906066</td>
<td>-75.353019</td>
</tr>
<tr>
<td>17</td>
<td>39.905799</td>
<td>-75.352462</td>
</tr>
<tr>
<td>20</td>
<td>39.905483</td>
<td>-75.351867</td>
</tr>
<tr>
<td>24</td>
<td>39.905555</td>
<td>-75.351119</td>
</tr>
<tr>
<td>32</td>
<td>39.906013</td>
<td>-75.350959</td>
</tr>
<tr>
<td>25</td>
<td>39.905578</td>
<td>-75.350814</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>19</td>
<td>39.905281</td>
<td>-75.351066</td>
</tr>
<tr>
<td>28</td>
<td>39.904907</td>
<td>-75.350959</td>
</tr>
<tr>
<td>25</td>
<td>39.90443</td>
<td>-75.350738</td>
</tr>
<tr>
<td>24</td>
<td>39.903785</td>
<td>-75.350791</td>
</tr>
<tr>
<td>20</td>
<td>39.903392</td>
<td>-75.351005</td>
</tr>
<tr>
<td>28</td>
<td>39.903728</td>
<td>-75.351524</td>
</tr>
<tr>
<td>28</td>
<td>39.904193</td>
<td>-75.352027</td>
</tr>
<tr>
<td>29</td>
<td>39.904228</td>
<td>-75.35276</td>
</tr>
<tr>
<td>18</td>
<td>39.903823</td>
<td>-75.353286</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>------------------</td>
</tr>
<tr>
<td>25</td>
<td>39.903171</td>
<td>-75.353408</td>
</tr>
<tr>
<td>29</td>
<td>39.902889</td>
<td>-75.353599</td>
</tr>
<tr>
<td>18</td>
<td>39.903022</td>
<td>-75.35437</td>
</tr>
<tr>
<td>20</td>
<td>39.903278</td>
<td>-75.355125</td>
</tr>
<tr>
<td>20</td>
<td>39.903457</td>
<td>-75.355667</td>
</tr>
<tr>
<td>26</td>
<td>39.903411</td>
<td>-75.357086</td>
</tr>
<tr>
<td>24</td>
<td>39.903644</td>
<td>-75.356597</td>
</tr>
<tr>
<td>19</td>
<td>39.903991</td>
<td>-75.356155</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td>18</td>
<td>39.904109</td>
<td>-75.355537</td>
</tr>
<tr>
<td>21</td>
<td>39.904373</td>
<td>-75.355186</td>
</tr>
<tr>
<td>21</td>
<td>39.904651</td>
<td>-75.354698</td>
</tr>
<tr>
<td>29</td>
<td>39.905044</td>
<td>-75.354866</td>
</tr>
<tr>
<td>20</td>
<td>39.905437</td>
<td>-75.354965</td>
</tr>
<tr>
<td>26</td>
<td>39.905883</td>
<td>-75.355339</td>
</tr>
<tr>
<td>30</td>
<td>39.906387</td>
<td>-75.355628</td>
</tr>
<tr>
<td>33</td>
<td>39.90681</td>
<td>-75.355216</td>
</tr>
<tr>
<td>22</td>
<td>39.907184</td>
<td>-75.355018</td>
</tr>
<tr>
<td>18</td>
<td>39.907512</td>
<td>-75.355377</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>15</td>
<td>39.907768</td>
<td>-75.355178</td>
</tr>
<tr>
<td>14</td>
<td>39.907512</td>
<td>-75.354591</td>
</tr>
<tr>
<td>17</td>
<td>39.907661</td>
<td>-75.354164</td>
</tr>
<tr>
<td>24</td>
<td>39.908245</td>
<td>-75.354064</td>
</tr>
<tr>
<td>33</td>
<td>39.908184</td>
<td>-75.35353</td>
</tr>
<tr>
<td>21</td>
<td>39.907684</td>
<td>-75.353546</td>
</tr>
<tr>
<td>28</td>
<td>39.907329</td>
<td>-75.353431</td>
</tr>
<tr>
<td>34</td>
<td>39.906784</td>
<td>-75.353347</td>
</tr>
<tr>
<td>28</td>
<td>39.906459</td>
<td>-75.353523</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>39.906017</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>39.905673</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>39.905654</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>39.906013</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>39.906524</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>39.906753</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>39.906753</td>
</tr>
</tbody>
</table>
Appendix G: MATLAB: Clean Up Data Rows Function
Filename: remove_NaN_rows.m

function mat=remove_NaN_rows(rawData)
    %Removes NaN columns from raw data
    L = length(rawData(:,1));
    for i=1:L
        if isnan(rawData(L-i+1,1))
            rawData(L-i+1,:)=[];
        end
    end
    mat=rawData;
end

Appendix H: MATLAB: Image Used for Creating Heat Map
FileName:May1_Picture.png
%% Import Data

% Import into arrays
May_1_Raw_Table = readtable('Data Collection.xlsx','Sheet','May 1','Range','A1:C188');
May_1_Raw = table2array(May_1_Raw_Table);

% Pull columns
May_1_Counts=May_1_Raw(:,1);
May_1_Lat=May_1_Raw(:,2);
May_1_Long=May_1_Raw(:,3);

% Clean rows
May_1_Counts_Clean = remove_NaN_rows(May_1_Counts);
May_1_Lat_Clean = remove_NaN_rows(May_1_Lat);
May_1_Long_Clean = remove_NaN_rows(May_1_Long);

%% GPS
close all

%create axis
n=100;
Lat_axis=linspace(min(May_1_Lat_Clean),max(May_1_Lat_Clean),n);
Long_axis=linspace(min(May_1_Long_Clean),max(May_1_Long_Clean),n);
gps_stdev=0.0001;

%create raw plot
%figure(1)
%plot3(May_1_Lat_Clean,May_1_Long_Clean,May_1_Counts_Clean);

alpha_efficiency = 0.0319; %efficiency
%Note: I changed alpha to alpha efficiency bc i needed alpha for surface
%plot

Error_alpha = 0.0562; % 95% confidence (2 std devs)
alpha_stdev = Error_alpha/2;
% create array with average values
processed_data = ones(n);
processed_data = processed_data * mean(May_1_Counts_Clean)/alpha_efficiency;

% Gaussian info
Gaussian_heights = (May_1_Counts_Clean-mean(May_1_Counts_Clean))/alpha_efficiency;
Gaussian_stdevs = May_1_Counts_Clean/alpha_stdev;

% pair each real world coordinate with one in the grid we created
Nearest_Lat = [];
Nearest_Long = [];
for i=1:length(May_1_Lat_Clean)
    Nearest_Lat = [Nearest_Lat Lat_axis(findNearest(Lat_axis,May_1_Lat_Clean(i)))];
    Nearest_Long = [Nearest_Long Long_axis(findNearest(Long_axis,May_1_Long_Clean(i)))];
end

% for ease of reference
[Lat_mesh, Long_mesh] = meshgrid(Lat_axis, Long_axis);

% adding in the Gaussians
for ind=1:n^2
    for jnd=1:length(Gaussian_heights)
        distance = pdist2([Lat_mesh(ind) Long_mesh(ind)], [Nearest_Lat(jnd) Nearest_Long(jnd)]);
        processed_data(ind) = processed_data(ind) +
        Gaussian_heights(jnd)*normpdf(distance,0,gps_stdev)/normpdf(0,0,gps_stdev);
        end
    end

% create surface plots

% creates peaks w/o map overlay
% figure (1)
% surface1=surf(Lat_mesh,Long_mesh,processed_data);

% creates peaks surface with map overlay
image=imread('May1_Picture.png'); %import photo/map
%imshow(image)
image=flipud(image); %define the flipped image as the color data for the surface
image = imrotate(image, 90); %play with the image to orient in right way

processed_data = flipud(processed_data);

figure (2)
surface2 = surf(Lat_mesh, Long_mesh, processed_data);
alpha (surface2, 0.5); % sets transparency to 25%

hold on

surface3 = surface(Lat_mesh, Long_mesh, 300*ones(n, n), image, 'facecolor', 'texturemap', 'edgecolor', 'none');
title('Radiation Levels of Swarthmore College')
ylabel('Longitude')
xlabel('Latitude')
zlabel('Radiation (Counts per Minute)')

c = colorbar;
c.Label.String = 'Counts per Minute';

figure (3)
surface4 = surf(Lat_mesh, Long_mesh, processed_data);
alpha (surface4, 0.5); % sets transparency to 25%
surface5 = surface(Lat_mesh, Long_mesh, processed_data, image, 'facecolor', 'texturemap', 'edgecolor', 'none');

% Note: this method requires a precise selection of map image to overlay
% In my test, the locations do not line up perfectly but I was able to
% overlay a map image with the surface plot

% Note: changed variable name of alpha when defining constants

% must adjust each image as desired to line up correctly as well

% https://www.mathworks.com/help/matlab/ref/surface.html
% also uses imtoolbox stuff