Basic DIY Gamma Spectroscopy
Using Better Geiger S-1 Radiation Detector and an Arduino Uno

www.bettergeiger.com

V1.1

March 2023

Please read everything before attempting this project!

List of contents

1. General context and warnings

This guide provides a very simple way to use the Better Geiger S-1 to make a basic DIY gamma spectrometer. This is not an officially supported use of the detector and the user attempts this project at their own risk. Mistakes can damage the detector and render it no longer functional. Such damaged units will not be repaired or replaced at the manufacturer's expense.

Furthermore, as this project uses the detector in ways beyond its intended design, the performance of any particular S-1 might not match the results described in this text. Specifically, the achievable resolution and dynamic range is highly variable from unit to unit. A detector designed for spectroscopy would normally be subject to extra calibration and quality control steps beyond what the Better Geiger S-1 undergoes for its intended dosimetry use. The electronics design of the S-1 is also not designed with spectroscopy in mind as a primary goal. This all has consequences and should be kept in mind to keep expectations reasonable.

Typical values are roughly 12-16% resolution (FWHM of 662 keV Cs-137 gamma) and an energy range which extends up to roughly 1000-1200 keV. There can be significant deviation from this "typical" range due to variability from one unit to the next. The setup described here is an early work in progress. There are likely bugs to fix and certainly a lot of places where improvement is possible. The goal was not to provide a polished product, but to get a minimum working setup to provide to people so they can get started experimenting on their own.

Anyone attempting this project should have some basic DIY electronics skills including soldering. They should also be familiar with the Arduino platform generally, including the Arduino Uno board and the

Arduino IDE. Knowledge assumed includes how to solder wires to the S-1 PCB and how to load a particular sketch onto an Arduino Uno. Material requirements are minimal – the Better Geiger S-1 itself, an Arduino Uno, a USB cable, a few pieces of wire, and a PC to read the data. To get the most out of the project, though, some different radioactive samples are needed. That is described in more detail later.

2. Project overview

The S-1 user manual describes "raw signal access" on the main PCB inside the enclosure. One signal is an analog pulse which corresponds to radiation interacting with the sensor. The amplitude of those pulses is roughly proportional to the energy deposited in the sensor for each individual radiation interaction, with some deviation from linearity. There is also a digital pulse which indicates that an interaction has occurred above a given energy threshold. The basic idea of this project is to connect three of those "raw signal" connection points to an Arduino Uno - the analog signal, the trigger, and the ground - to record gamma spectrum data. Additionally, the 5V from the Uno can optionally be connected to the S-1 in order to power it (only with batteries removed!), otherwise normal battery power can be used instead.

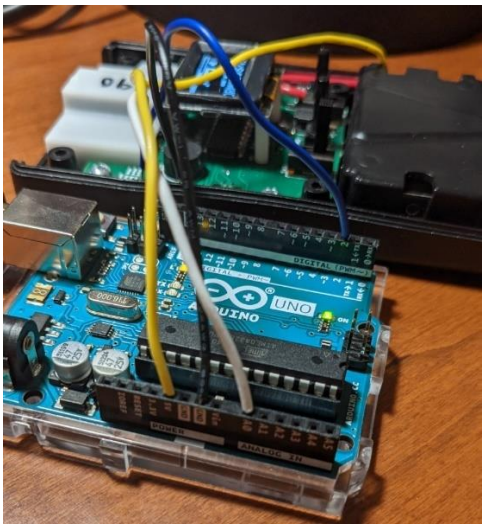
Basically, the trigger tells the Arduino "radiation has interacted with the sensor", and then the Arduino quickly reads out the amplitude and saves it. By recording many of these events over time, a spectrum can be generated. A gamma spectrum is basically a histogram of pulse heights. The X axis is pulse height (roughly corresponds to energy), and the Y axis is number of events at each pulse height bin. Where there is a peak or a hump in the spectrum it indicates that a lot of events around that energy have been recorded. This can be seen in the results shown later.

The basic setup described in this document is with the Arduino Uno connected to a PC by a USB cable. The Arduino IDE is used to load the "sketch", the code that the Arduino runs, and the Arduino IDE can also be used to read the serial port data sent by the Uno (data it is collecting from the S-1). The data, a text list of 128 numbers, can then be copied into Excel, Google sheets, or any other program suitable for visualization and further analysis. Each number in the spectrum data corresponds to events counted in a given bin, starting from the 1st bin and going up to the 128th. Going further along the list of numbers means going to higher amplitudes corresponding to higher energies.

3. Setup of connections

The first step is to remove the top of the S-1 case, held in place by four screws on the back side. Two of them are under the battery cover. The raw signal access connection points should then be visible to the "left" of the display. If powering from the Arduino 5V output then this would go at any point along the red wire connected to the battery terminals (batteries should be removed if providing external power!). The connection can be implemented in a variety of ways but simple hook-up wires soldered to the S-1 PCB is probably the easiest. The hole above the buzzer can be enlarged with a drill to allow running the wires through that hole when the case is closed. When closing the case, care should be taken that the display holes are properly centered on the white spacer piece, and that the button and switches are carefully aligned while closing it up to avoid damage to any components.

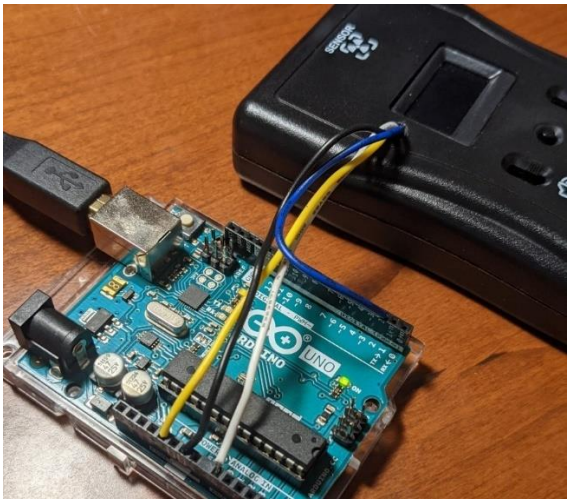
Overall image of setup with enclosure open:



Three raw signal connections soldered:

Optional 5V connection **(remove batteries!)**

Final setup with wires run through an enlarged hole above the buzzer:



4. Arduino Sketch

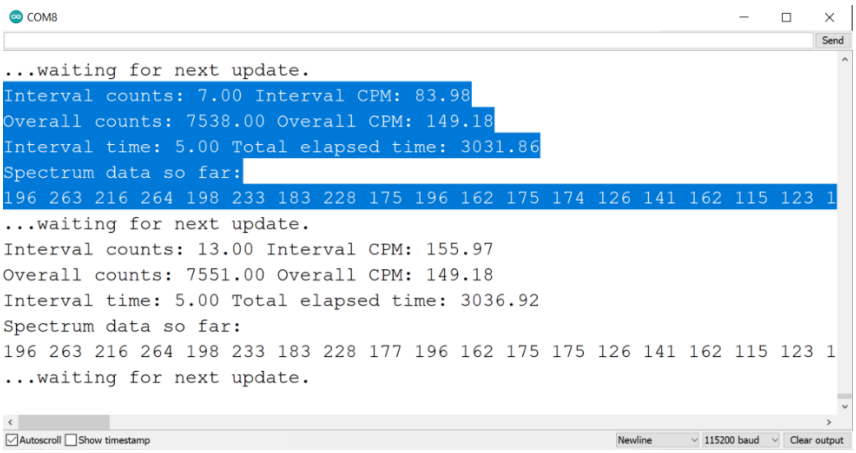
Below is the Arduino sketch inblue. It is about 100 lines of code and can be copied into the Arduino IDE, saved, and loaded to an Arduino Uno.

```
//
// GENERAL INFO ABOUT SKETCH
//
// Better Geiger S-1 - Basic spectroscopy setup using an Arduino Uno
// FULL DETAILS AND INSTRUCTIONS: SEE www.bettergeiger.com and go to "User Documents"
// This is sketch version 1.0
//
// This sketch intends to allow basic extraction of spectrum data from a Better Geiger S-1
// This setup is suitable to use with all detectors shipped prior to May 2023
// The current design (up to April 2023) has an analog signal width of 60 microseconds.
// The code here, used with an Arduino Uno, reads the signal within about 50 microseconds.
// No changes have been made yet but future S-1 firmware (later 2023) might have a faster
// analog signal which would mean a faster readout than this code achieves would be necessary
// for it to work. If uncertain, it is best to check the analog signal with an oscilloscope.
//
// If using a different board than Arduino Uno, note the following:
// 1. the three lines commented in the code "Uno only" probably need to be removed
// 2. the analogRead must occur in about 50 microseconds or less with current S-1 design

//
// BEGINNING OF SKETCH CODE
//

// code used for analog read speed increase
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit)) // Uno only
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))  // Uno only

unsigned long  intervalStartTime;      // start time for a given data recording interval [ms]
unsigned long  intervalElapsedTime;    // amount of time elapsed for a given interval [ms]
unsigned long  overallStartTime;       // overall start time [ms]
unsigned long  overallElapsedTime = 0;  // overall start time since device turned on [ms]
double         intervalCounts     = 0;  // number of counts during given interval [-]
double         overallCounts      = 0;  // counts since device was turned on [-]
double         overallCPM         = 0;  // counts per minute since device was turned on [counts/min]
double         intervalCPM        = 0;  // counts per minute during interval [counts/min]
volatile int   H;                    // analog pulse height [ADU]
int          H_old = 0;              // variable used to check for change in pulse height [ADU]
bool           exitWhileLoop = false;   // variable to track when exit of while loop is needed [-]
int          spec[128];              // pulse height spectrum data array
int          bin;                  // integer for bin position index

// setup section

void setup() {
 Serial.begin(115200); // start serial communication
 overallStartTime  = millis(); // get overall start time
 Serial.println("Starting."); // print to Serial port that the device is starting
 attachInterrupt(digitalPinToInterrupt(2), registerCount, RISING); // define trigger interrupt
 // increase analog read speed 4x compared to default
 sbi(ADCSRA, ADPS2);   cbi(ADCSRA, ADPS1);   sbi(ADCSRA, ADPS0); // Uno only
}

// main operating loop

void loop() {
```

```
 exitWhileLoop = false; // while loop below will run until this is set to true
 intervalStartTime = millis(); // get the start time for this measurement interval
 intervalCounts = 0; // start with 0 counts in this measurement interval
 while (exitWhileLoop == false) { // run until exit loop variable is changed
  intervalElapsedTime = millis() - intervalStartTime; // get current interval elapsed time
  if (intervalElapsedTime > 5000) { // check if interval duration has been reached (default 5000 = 5 s)
    exitWhileLoop = true; // if yes then end this counting interval
  }
  if (H != H_old) { // check if H has changed
    H_old = H; // if so then reset the "old" value to be the current one
    if (H > 1) { // this ignores spurious zero values
      bin = H * 0.125; // down-samples from the default analogRead down to 128 bins
      spec[bin] = spec[bin] + 1; // add one to the appropriate bin value
      intervalCounts = intervalCounts + 1; // add one to the interval event counter
    }
  }
 }

 overallElapsedTime  = millis() - overallStartTime; // get overall elapsed time
 overallCounts = overallCounts + intervalCounts; // add interval counts value to overall counts value
 intervalCPM = intervalCounts / intervalElapsedTime * 1000.0 * 60.0; // calculate interval CPM
 overallCPM = overallCounts / overallElapsedTime * 1000.0 * 60.0 ; // calculate overall CPM

 // print general info to serial port
 Serial.print("Interval counts: ");    Serial.print(intervalCounts);
 Serial.print(" Interval CPM: ");     Serial.println(intervalCPM);
 Serial.print("Overall counts: ");     Serial.print(overallCounts);
 Serial.print(" Overall CPM: ");      Serial.println(overallCPM);
 Serial.print("Interval time [s]: ");      Serial.print(intervalElapsedTime/1000.0);
 Serial.print(" Total elapsed time [s]: "); Serial.println(overallElapsedTime/1000.0);
 Serial.println("Spectrum data so far:");

 // print spectrum data to serial port
 for (int i = 0; i < 128; i++){Serial.print(spec[i], DEC); Serial.print(" ");}
 Serial.println(""); // move to next line
 Serial.println("...waiting for next update."); // print that this interval is concluded
 Serial.println(""); // skip a line before next set of data is printed

}

// Interrupt routine
// this enables that when the trigger comes, the analog signal amplitude is recorded
void registerCount() {
  H = analogRead(A0) ; // read analog value
}
```

5. Basic visualization of gamma spectra

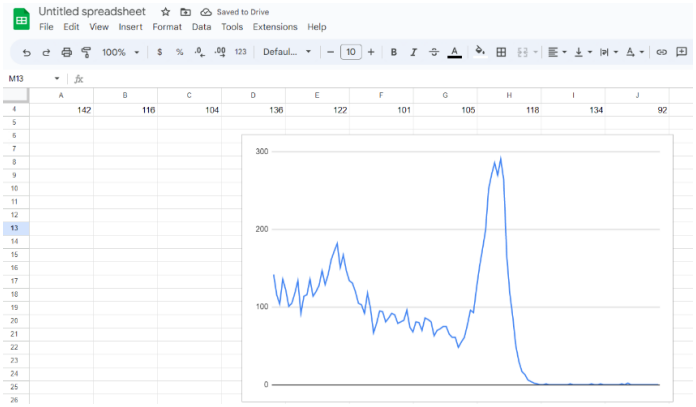
After the physical connections are set up and the sketch is loaded onto the Arduino Uno, by using the Arduino IDE to monitor the serial port it is possible to view and record basic measurement info along with spectrum data. The simplest way to deal with the data is to copy some text from the terminal into some other program. That might look something like what is shown below, with the blue text highlighted to be copied. The most critical information includes the total elapsed time and the spectrum data itself on the long line.



The spectrum can be processed/visualized further with a wide variety of programs, but Google Sheets is described here as a simple and free example. The copied spectrum line will all be put in one cell at first, and it can be split into individual cells by selecting the cell with the full line of data, then going to “data” and “split text into columns”, then selecting “space” as the separator. Then by highlighting the line and clicking “Insert chart” the measured spectrum can be seen. Examples are given in the following section.

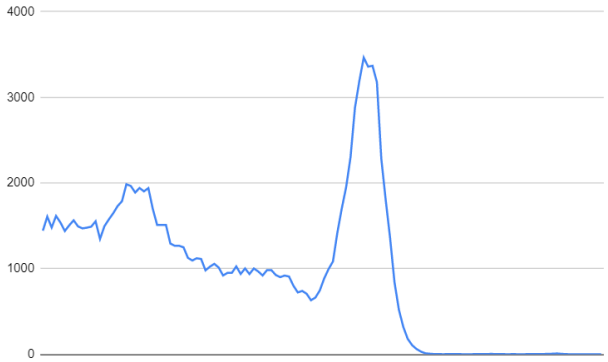
6. Understanding gamma spectra through examples

Below is a quick measurement of a Cs-137 gamma source generated using the previously described visualization steps. The Google Sheets context is shown.



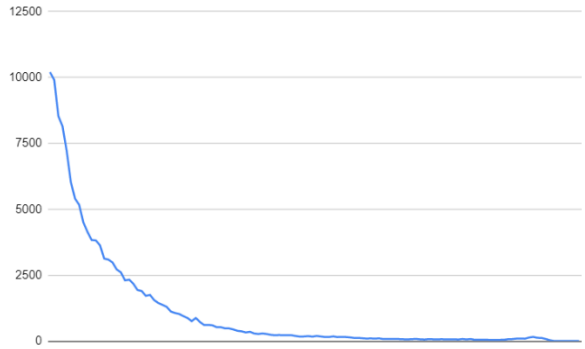
On the horizontal axis is each measured bin, 1 to 128, with higher bin number corresponding to higher energy. The height of the line at each bin indicates how many counts were registered in that bin. On the right there is a peak which corresponds to the 662 keV gamma emission of Cs-137. Even though Cs-137 is emitting just one energy, and most of what is interacting in the detector is from that energy, not all events deposit exactly 662 keV. Sometimes there is full energy deposition, and that corresponds to the peak. The peak is more of a hump than a perfectly sharp line because there is noise and uncertainty causing it to be blurred out in that way. There are also many events to the left of the peak recorded, those correspond to Compton scattering or other events in which the full 662 keV was not deposited and recorded. The basic shape is typical for sources that emit one prominent gamma energy, one main peak and a lower plateau to the left which might have some secondary (not very meaningful) structures that could be confused with true peaks. Interpreting gamma spectra is a complicated topic, some basics concepts are given here but full understanding is far beyond the scope of this document.

Another point to note is how jagged and jumpy the line is. That's because this was a very quick measurement. The bins around the peak location have up to roughly 150 counts in each bin, as can be seen on the y axis. The standard deviation for a given bin is generally the square root of the number of counts. For 150 that means 12.2 counts or, as a percent, 12.2/150 = 8.1%. That is a very high uncertainty for each bin and explains why the line is very jumpy. It takes much larger numbers of counts per bin to reduce the "noise" in the results. For comparison, next is shown the same measurement setup but with longer measuring time:



Below is a longer measurement of Uranium ore. There is a small peak visible towards the middle near where the 662 keV peak was previously seen, but it is not from Cs-137 - it corresponds to a 609 keV gamma emission that is significant in the Uranium decay chain. Uranium ore contains not only uranium but a series of isotopes which it decays into. There are many low energy emissions and those seem to be blurred together on the left side due to the limited resolution of the detector. On the far right there seems to be two small peaks. The one on the left might correspond to another emission in the decay chain. The one on the right contains anything which is too high for the range to resolve it properly, because it gets pushed into the upper end. In other words, that is probably not a true peak but just a result of the clipped upper range, and this is a typical artifact when the measured spectrum extends beyond the measurable energy range. Even though there are many counts per bin in the lower energy range, a noticeable jaggedness is still apparent. This is probably not due to counting statistics in this case, but some artifacts of the crude manner in which the signal is being handled using an Arduino Uno. If searching for very minor structures in a spectrum this might cause complications.

Shown next is a background spectrum with no particularly radioactive sample nearby. This device is not capable of resolving any noteworthy structures in the background spectrum, but it can be seen that the bulk of the spectrum is mostly towards the lower energy range.



A background spectrum is important to measure if a sample being tested does not produce a high count rate. For example, if the background count rate is 100 CPM and a sample measurement has a count rate of 200 CPM, only 100 CPM is from the sample itself and only half of the "with sample" measurement will actually be from the sample, with the rest just being distortion from the background. If the sample is producing perhaps 10x or more count rate compared to the background this is a minor distortion that might not be worth correcting, but for weaker samples it is worth doing two measurements, one with and one without sample. If they are measured for the same amount of time the background spectrum values can be subtracted from the sample spectrum values one to one. If they are for different measuring times, then they need to be scaled accordingly. For example, if the sample measurement was 1 hour and the background were 3 hours, the 3 hour spectrum values need to be divided by 3 before subtracting from the sample. These kinds of operations can worsen the counting statistics noise due to how error propagation behaves. This is why it is common to measure for long periods of time to get good data, often for many hours. The scaling issue when subtracting one from another is also why recording total measuring time is important when saving spectrum data. Good notes are also valuable, including sample information, where it was positioned on the detector, and any other noteworthy details of the setup. Photos are also a good idea to remember what was measured and how.

7. Ideas for measurements and general tips

One of the simplest ways to test that a measurement setup is working and to get some initial results is with a Uranium ore sample, available for purchase from https://www.bettergeiger.com/product-list but with US shipping only. With a Uranium ore sample and a background measurement a person can check to see that the spectroscopy setup is giving reasonable results. Natural minerals containing thorium are also of interest but harder to find, but with luck can be found when sold by vendors aiming at rock collectors – searching online might yield results.

Radioisotopic sources, such as the previously shown Cs-137, can be purchased online as well. The supplier "Spectrum Techniques" dominates the hobbyist-grade source market and there are a variety of sources available for roughly $50-100 and up each.

Uranium glass and Fiestaware can be tried but they are likely to be too weak to produce very interesting spectra. The quantity of uranium is low and what is emitted is primarily low energy beta not suitable for this device to record any meaningful spectrum.

Some types of smoke detectors contain an Am-241 radioactive “button” inside. Some hobbyists extract this button to use as an artificial radiation source but legality should be verified before attempting that. Am-241 emits (among other things) a 60 keV gamma ray that the detector will react to strongly, but will not produce a meaningful peak because it is so close to the lower detection threshold of 50 keV.

Potassium chloride exists in water softeners and other commercial products and is another common source of radioactivity in everyday life. The potassium emits a high energy gamma that may be beyond the upper range of the S-1, depending on which particular unit is used.

Some granite countertops contain significant quantities of potassium or other naturally radioactive materials. The potassium, as previously mentioned, emits a rather high energy gamma, but when measuring a bulky distributed emitter like that the spectrum might be very blurred out towards lower energies due to scattering within the granite itself.

Thorium is present in some objects, one of them being thoriated tungsten welding rods. Such rods are readily available for purchase and a pack of 10 or so bunched up and placed next to the sensor can produce a gamma spectrum.

Generally speaking the higher the count rate the better as long as an extremely high rate is not reached. At extremely high rates, roughly 20,000 CPM and above, the signals will start to pile-up noticeably and potentially distort the spectrum. In that case moving the sample a bit further away to avoid this pile-up is a good idea. This is unlikely to be a problem, though, except for artificial sources of particularly high strength.

Aside from those extremely high rate scenarios, generally it is best to have the sample as close as possible to the detector in order to achieve as high a rate as possible. This will allow faster acquisition of data and higher quality results. That usually means placing the sample directly on top of the “sensor” marked position on the detector, or directly under it, or at the end.

8. Ideas for further development of the setup

There is an enormous range of options for people who want to develop the general approach outlined here to have more features or better performance. Aside from manually copying and pasting data from the Arduino IDE, any number of software approaches could be applied to automatically pull data from the serial port, save it, and display it. Scaling of the bins to actual energy could be attempted using known-energy sources, but this might have a limited accuracy due to some degree of non-linearity in the energy vs. pulse height, so care should be taken.

The Arduino Uno was chosen because it is very widely available, but there are many faster options available that might allow higher quality sampling of the signal. Some have on-board data storage to avoid the tethered and cumbersome serial port data transfer approach. A display could be used to show the spectrum data live. A battery-powered setup could allow mobile data collection.

The core functionality described in this document could be built into any number of different arrangements for use in a variety of hobby projects.