DIY Geiger Counter Instructions

In this manual we describe the operation of and provide the assembly instructions for the MIT Laboratory of Applied Physics (LANPh) Geiger-Muller Counter. In order not to repeat ourselves too much, please refer to the following README on our github: https://github.com/ustajan/Geiger_lanph/tree/main

The Geiger counter is a type of particle detector. It can detect hits from gammas – essentially ~MeV photons that come from nuclear decay in various radioactive materials that surround us. There is an abundance of radioactive materials in our environment. These include ^{40}K (a naturally occurring isotope of potassium – a chemical that's fundamental to the operation of our brains), uranium and thorium (naturally occurring in the soil and granite), etc.. The counter is also sensitive to cosmic muons – which are produced in the upper layers of Earth's atmosphere due to bombardment by highly energetic cosmic rays. Armenia in particular has elevated rates of natural radiation due to its volcanic environment and high altitude.

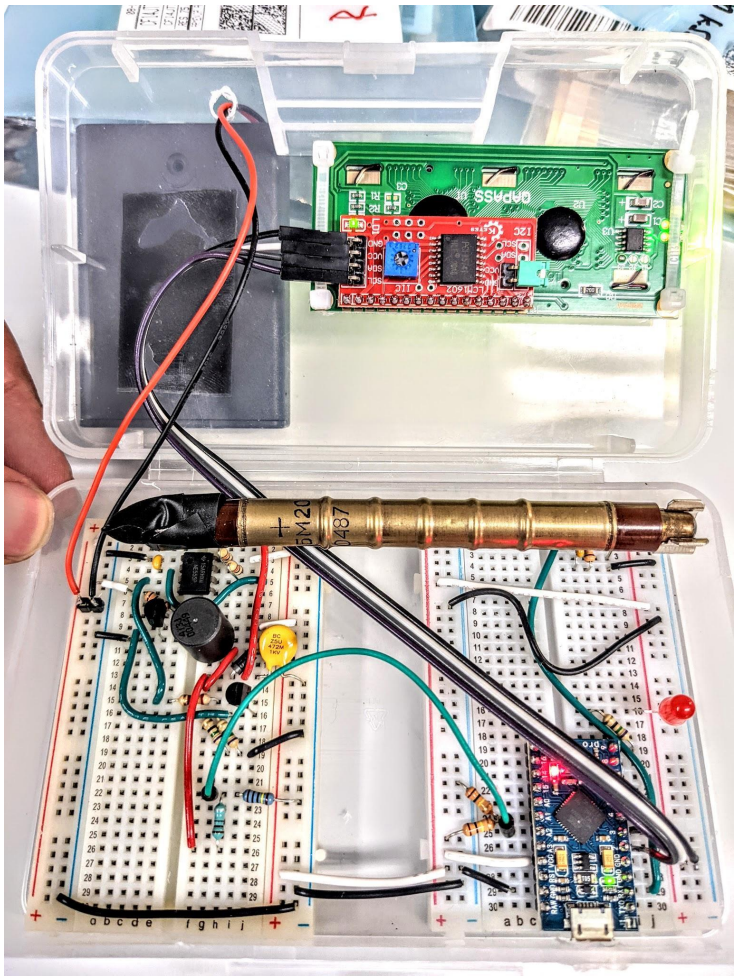
Assembling the Geiger counter is a great way to learn about electronics, engineering, and nuclear physics. It is also a great way of achieving a basic understanding of the nuclear processes that surround us – and have surrounded us since the inception of life 3 billion years ago! It is especially relevant for Armenia, given its higher-than-usual background radiation, as well as its use of nuclear power to provide 30-40% of its energy.

The students who take part in this two-week long course will walk away with an instrument that they've built with their own hands which they can then use to measure and map radiation in their communities and surroundings. They will also have a deeper understanding of what radiation is, where it is, and how it affects life. As part of the course they will learn the basics of nuclear physics, radiation detections, and basics of statistics and data analysis. They will build the Geiger counter and use it to make measurements of various radioactive sources and study the effects of shielding on radiation.

Geiger Counter – what is it?

At its core the Geiger Counter contains an ionization chamber (referred to as a Geiger tube) which is subjected to high voltage (HV) of ~300 Volts. When particles interact with the gas in the chamber they cause ionization, which then results in an avalanche of electrons from the cathode to the anode. This results in a current that is detected as a pulse in the voltage on the negative terminal. The pulse is then sent to an Arduino microcontroller which detects it and performs the basic analysis – showing the count rate and estimating the surrounding radiation dose.

The rest of the Geiger Counter is made up of the analog electronics to produce the HV and to process the initial current pulse. Below are the images of the assembled device.



The Design

While there are many designs for Geiger counters out there, this design is based on the following objectives:

- Simplicity. No / minimal soldering. Built from fairly common electronic and hardware components.
- Easy to hack. Easy to assemble. Easy to make quick changes. Hence the breadboard and not a PCB.
- Low price. If the components are purchased in bulk the total unit cost is approximately $52 -- this includes the Arduino, LCD, and the Geiger tube.
- The design is not meant to be a product. It is not meant to be pretty. It is instead meant to be an educational platform to help students learn about
    - nuclear detection, e.g. by performing experiments with various radioactive sources
    - statistics, e.g. by collecting data and observing various nuances of Poisson processes
    - the basics of analog electrical design
    - the basics of Arduino-based digital and software design

The design can be split into three main parts: HV module; Radiation sensor; DAQ and Readout.

HV module

The HV module takes the 4.5 V from a battery pack (or 5V from USB interface) and boost converts it to ~300V.

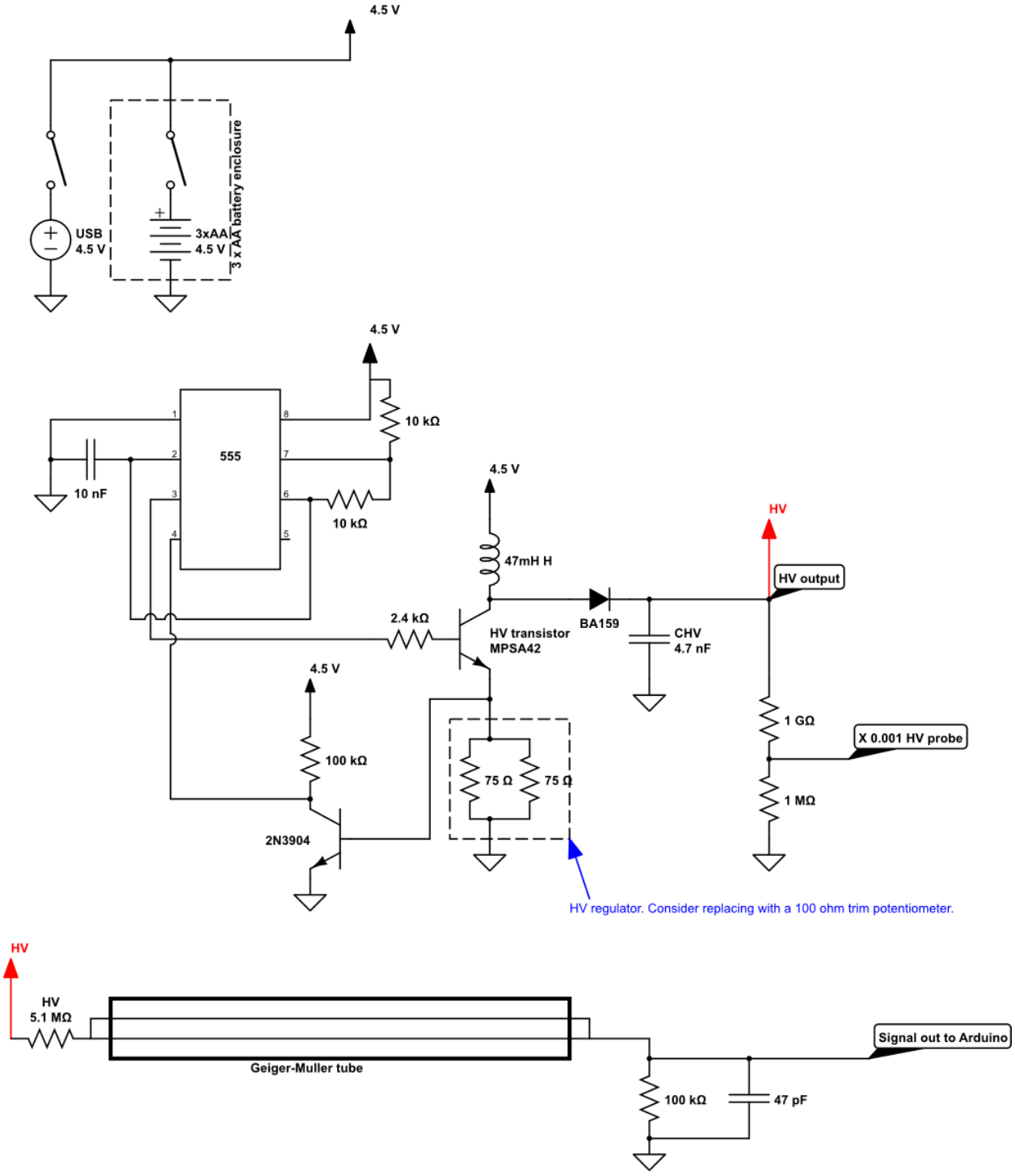
This is achieved by using a 555 timer as a switching device in feedback mode. The "core" of the HV is an inductor coupled to a capacitor through a rectifier diode. To achieve the HV the inductor-ground connection is "switched" on and off via a HV transistor, which takes the switching input from the 555 into its base. This is the basic principle of how spark plugs on an internal combustion engine work. It exploits the phenomenon of induction to produce a high voltage. Remember, the voltage across an inductor is $V = L\, dI/dt$…if you change I rapidly you get a large V.

Radiation Sensor

The HV is then used to bias the Geiger-Muller tube, which in our case is the classic Soviet SBM-20 ($20 on ebay) through a 5.1 MOhm input resistor. The negative terminal

of the tube is sent to ground via a 100 kOhm resistor. The signal is read directly from the negative terminal.

The diagram representing the analog part of the design can be seen below.



Data Acquisition (DAQ)

The DAQ is based on a small Arduino microcontroller. It takes the output directly from the Geiger tube into one of its digital pins.

The digital input requires a TTL signal, and while the output of the Geiger tube is not exactly TTL it's close enough to achieve reliable triggering. The Arduino sketch can be found on one of our other repositories here.

For any DAQ system the key question to ask is -- what is the dead time? We estimate that the dead time for the DAQ configuration employed in this design is <1ms.

The DAQ performs the following actions:

- estimates the count rate via a variable integration window and displays it on a 16x2 standard LCD. The integration time is determined as following:
    - When the observed rate is <20 CPM the integration time is 30 sec
    - else, if its <100 CPM the integration time is 10 sec
    - else: 1 second
- In addition to the count rate the DAQ also displays the dose rate in uR/hr. A few points:
    - The conversion coefficient for SBM-20 is 0.57 (uR/hr)/CPM.
    - Note that this conversion is based on 137Cs standard, i.e. 662 keV. This conversion may be completely inaccurate for lower energy sources. To "flatten" the energy dependence the tube can be wrapped in ~mm of lead as a way of achieving what's known as energy compensation.

Readout to computer via the serial connection

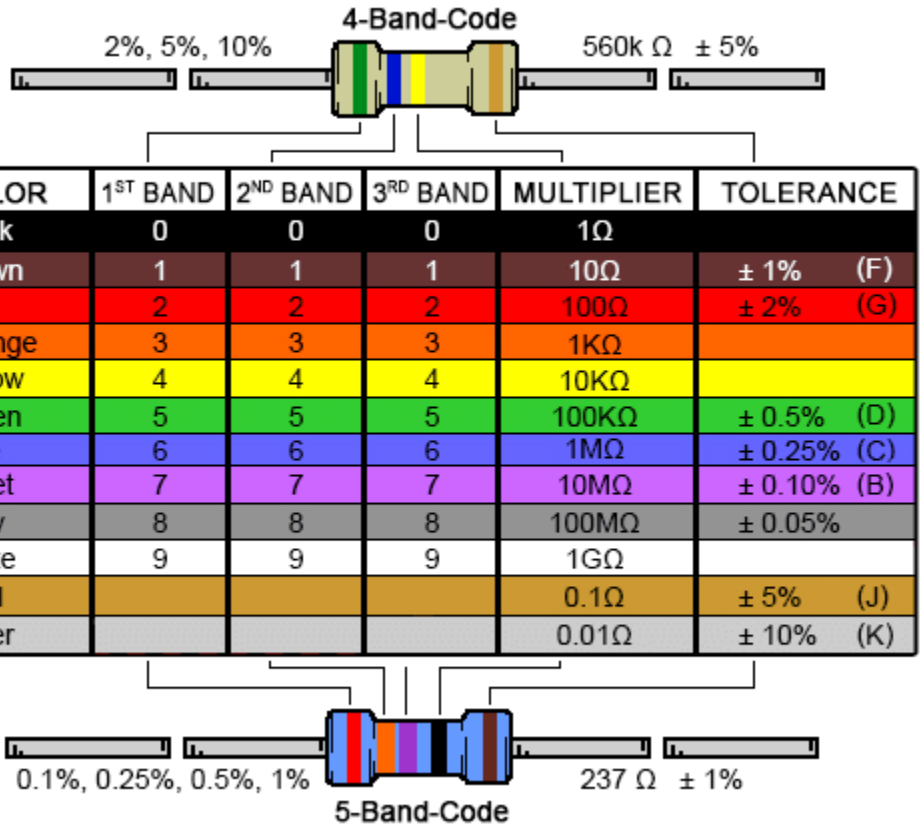
In addition to analyzing the count and dose rates and displaying those on the LCD, the Arduino DAQ software can also "talk" to a computer via the serial bus. The software for this can be found in the above mentioned repository here -- look for `logger.py` python code. The code reads and prints to file the timestamp of every event in microseconds, allowing for a subsequent detailed statistical analysis.

Instructions of Assembly

Below are the step by step instructions for assembling the device. For all the steps we suggest that you have a printouts of the following on the table next to you:

- schematic diagram from the above pages
- printout of the resistor color code
- Printout of the instruction manual

Having those next to you will make the steps easier to follow.
Here's the resistor color code:



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8	100MΩ	± 0.05%
White	9	9	9	1GΩ	
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

(from https://www.digikey.com/-/media/Images/Marketing/Resources/Calculator/resistor-color-chart.png)

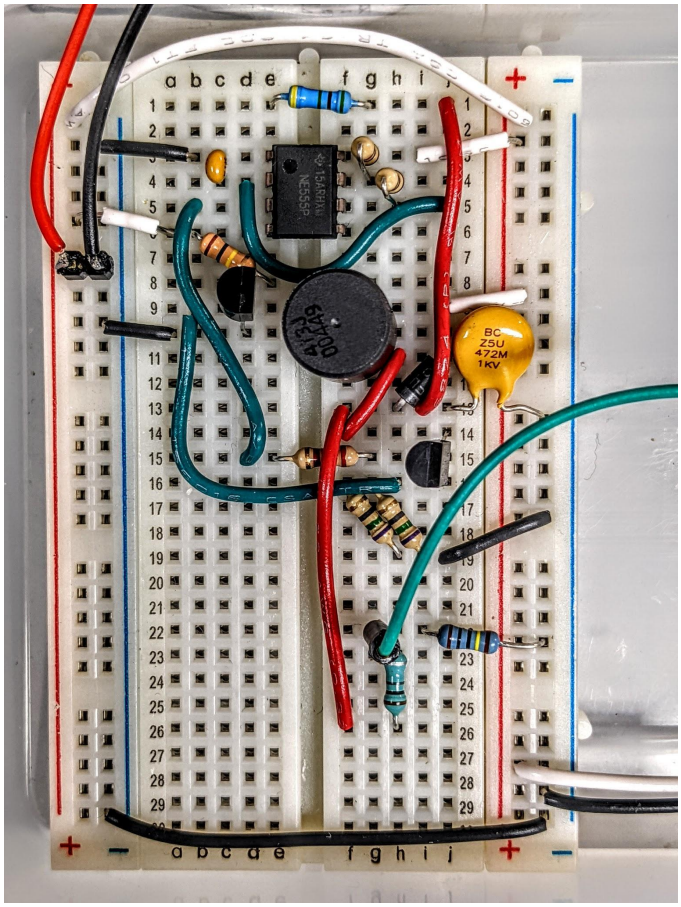
Step 1a: assembling HV

The first step is to assemble and test the HV module. This module takes the 4.5 V input and boost converts it to 300-400 V. For this step we suggest that you place the first breadboard(**bread0**) on the table and work that way.
A few suggestions for making your work easier and for preventing a messy outcome:

- Trip the components' leads to the length necessary, that way they are placed flat agains the breadboard

- For the resistors bend their legs in a square form so that they too are flat agains the surface. This can be tricky as you need to achieve the correct distance between the relevant breadboard cells. Ask for help from the instructor if this is not clear.
- You'll need to use wires to make various interconnects. For ease of following your own work we suggest the following four colors:
    - **Blue** – ground
    - **White** – 4.5 V
    - **Red** – HV line
    - **Green** or **black** – internal signal interconnections

Below is an image of the assembled HV module on the first breadboard (**bread0**).



You are not yet done with HV. You need to power it with 4.5 V! Here are the steps for electrifying your HV:

- Take your battery pack and load it with 3 AA batteries. Make sure that the switch is in the OFF position!
- Solder a 2-pin header to the leads. It's best to insert the 2-pin header into the power bus (the blue and red lines on the sides of the breadboard) and THEN do the soldering.

Step 1b: test the HV

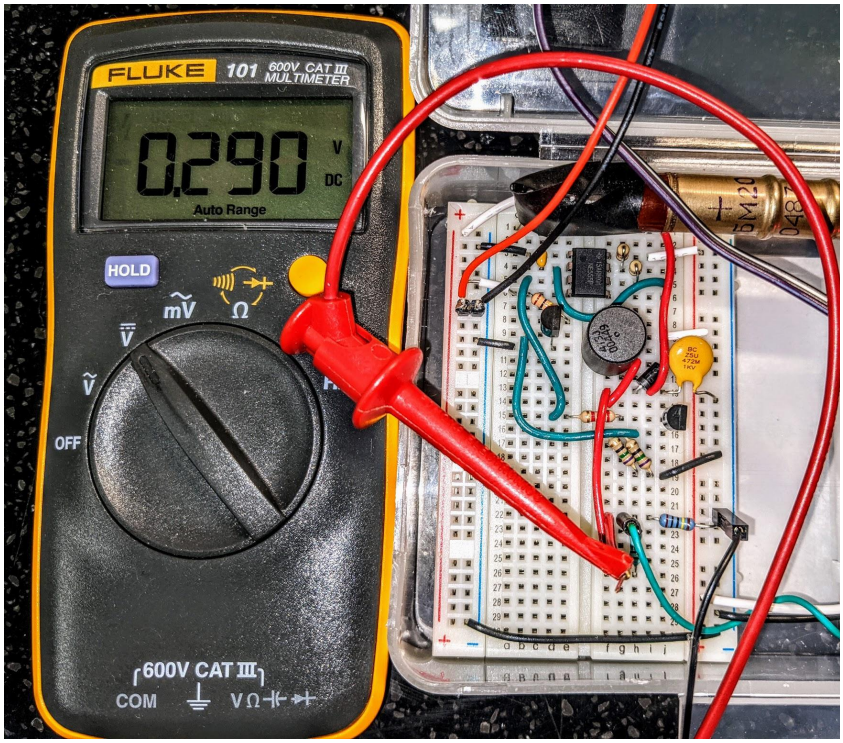
This is the fun (and also a little dangerous) part of your work! Flip the switch on the battery pack.
Flip the switch on the battery pack to ON. If your circuit is assembled correctly you'll immediately hear a high pitched ultrasound noise (this sound will be heard better by young people – why?). If you don't hear it then stop and go back to your circuit and check the various components.
If you do hear the noise then your HV is on and it's time to check the HV value!

Checking HV is tricky for a number of reasons – one being **safety**! While our HV module doesn't produce enough current to kill you it has enough energy to give you an unpleasant shock if you touch the wrong thing. This is why we have built a voltage divider that divides the HV by a factor of 1000 – that way you measure not 300V but 0.3 V instead. That's much **safer**!

On the circuit diagram on p4 you can see the contact that's marked "**x 0.001 HV probe**." We want to measure from there and then multiply the observed value by 1000. On the image on p7 the connection is on **row 22-right (bread0-22-right)**.
So let's do it!

- Turn the battery pack OFF
- Hook up a voltmeter to the row **bread0-22-right** (or whatever it is on your breadboard), while connecting its ground to the ground on the breadboard
- Turn the battery pack ON

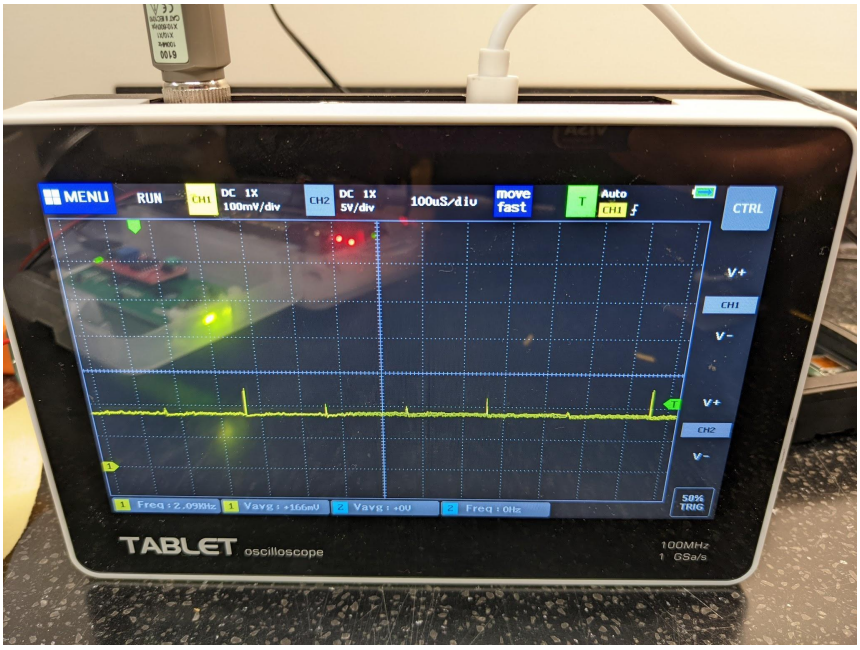
The voltmeter should measure something in the range of 0.29-0.35 V.



If your voltmeter has an internal impedance of 10 MOhms then you actually need to multiply this value by **x1100** to get the actual HV value. If the result you get is not 300-350 V then consult the instructor: you might have to adjust the feedback to the 555 timer by changing resistances in row 16-right.

Step 1c: observing the HV signal on an oscilloscope (**optional**)

You can also look at the (1000x reduced) HV output on an oscilloscope. Hook up your oscilloscope probe to the same exact place where you had the voltmeter connected. You should see something like this:



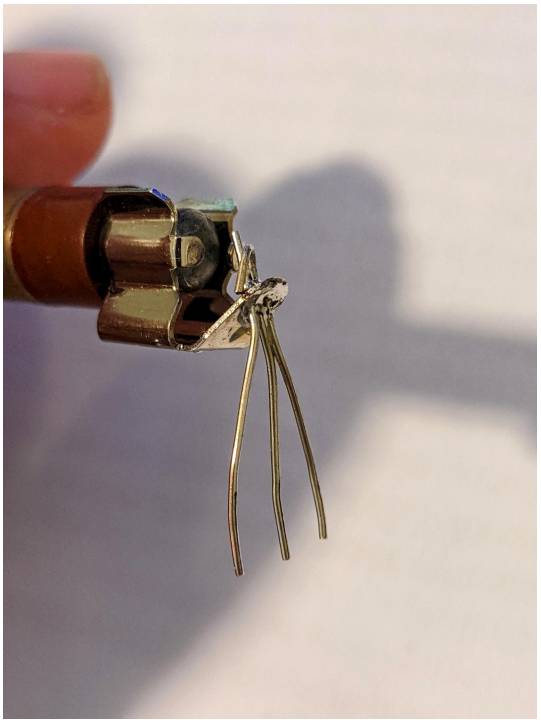
What do you think is happening? What are those spikes on the otherwise flat line? Remember we used to talk about "**switching**?" Can you think of ways to get rid of them?

Also note that the flat line is not at 0.290 V (as was the case for the voltmeter), but at 0.180 V instead – why is that? **Hint**: input impedances!

Step 2: resistor and capacitor for grounding the Geiger tube

This step just involves adding the 100 kOhm resistor and the 47 pF capacitors that are necessary to read out the Geiger tube. On the pictures in p7 and p8 they are located **on bread1-1-right (i.e. 2nd, right breadboard)**.

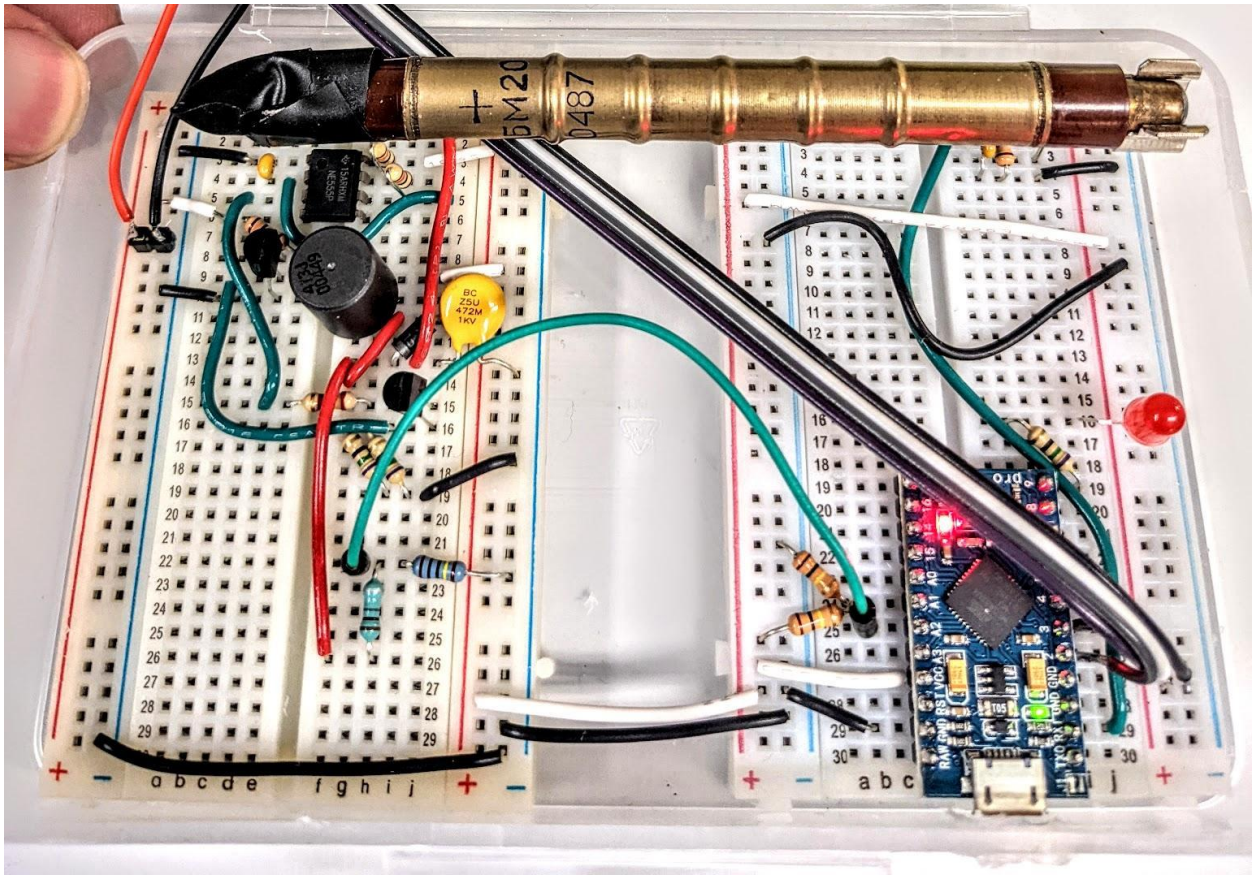
Step 3a: mounting the Geiger tube

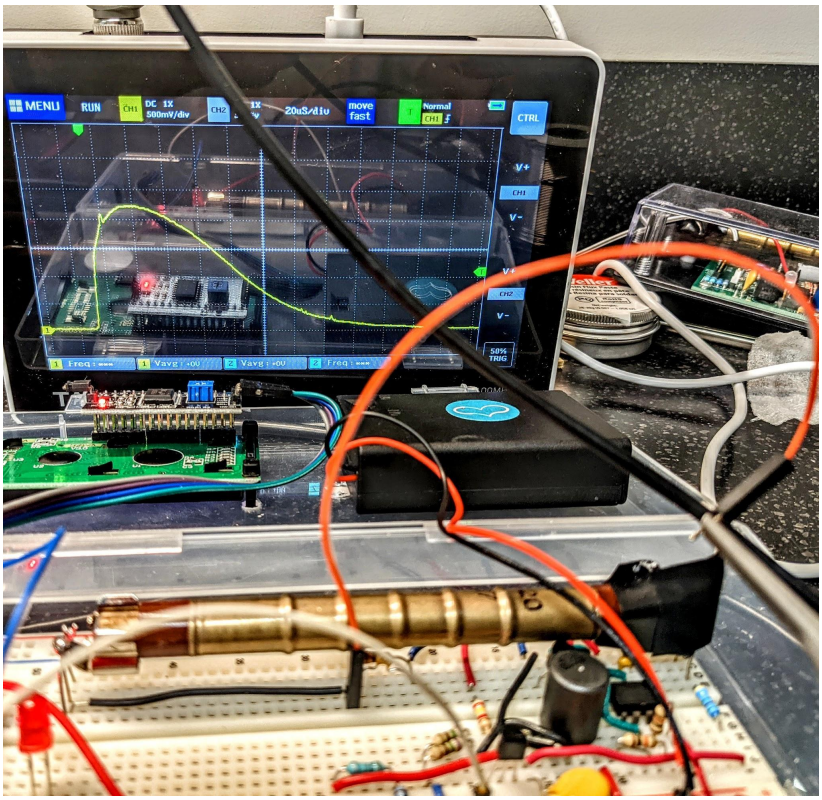


We now have to mount and connect the Geiger tube. For this you first need to solder THREE bare wires to the clips that connect to the terminals of the tube. See the picture on the left.
Do this for BOTH terminals. Additionally we strongly recommend that after you install the clip on the tube terminals you use electrical tape to wrap the **positive** terminal. This will help you avoid electroshocks later!

Finally, the last part! Now that your Geiger tube has both clips mounted on it, you can go ahead and mount it on the breadboard. Positive (HV) terminal needs to go into row **bread0-1-left**, and Negative terminal needs to go into row **bread1-1-right**. You also need an interconnect from your 100 kOhm resistor to the negative terminal of the tube. Here's the fully assembled thing. Notice that the two breadboards (bread0 and bread1) also have power lines connecting their rails.

Also, the rails on both sides of the breadboards are interconnected with **black** and **white** wires.





Step 3b: testing the Geiger counter

Finally it's time to test the analog part of the device! Connect an oscilloscope to the top of the 100 kOhm resistor (**bread1-1-right**) and turn on the device. If everything is working you should see nice, juicy 2V pulses on the screen. You should expect a count every few seconds.

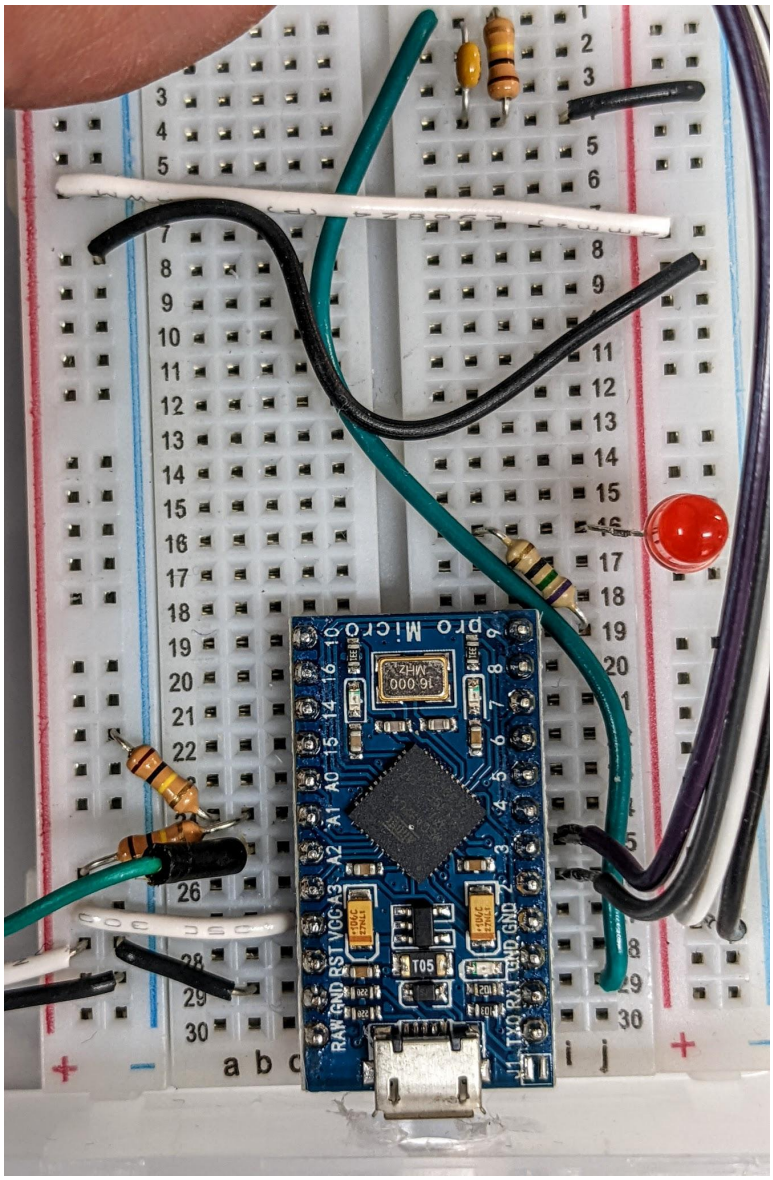
Note that your pulses have little blips on them : that's the HV switching. Not a big deal – it doesn't affect our measurement.

(Also note that this image uses a single breadboard – this is an older design. The rest of the idea is the same).

Step 4: instrumenting the Geiger Counter with Arduino DAQ

Congratulations!! At this point you have a working analog Geiger Counter – time to take a short break and enjoy your accomplishment!

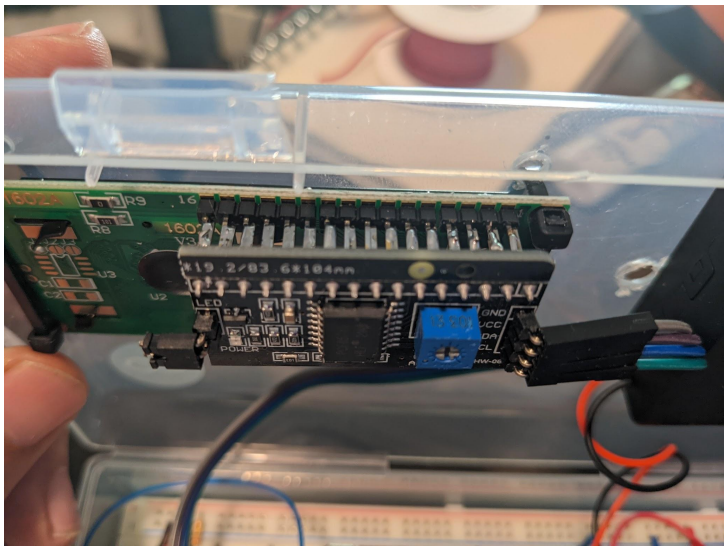
Once you are back from your break we can now work on adding an Arduino microcontroller to digitize and process the analog signal from your Geiger tube. Due to its relative simplicity, we do not have a schematic drawing of the Arduino interconnects. Instead here's a photo that describes the general setup. Note that here we use a Pro Micro. For other chips, e.g. Arduino Nano Every, some of the interconnects may be different – in particular the connection to the I2C of the LCD are different.



Here are some explanations:

1. The signal from the 100 kOhm resistor is sent to pin **RX1** – this is digital pin 0. On Arduino Nano Every digital 0 corresponds to the label of **RX0**.
2. Pin 9 operates the red LED – which blinks every time there is a detection
3. Voltage measurements:
    a. Analog-2, marked **A2**, reads the voltage from the voltage divider (the **green** wire goes to **bread0-22-right**)
    b. Analog-1, marked **A1**, reads the battery voltage, i.e. the common voltage for the whole device. The two resistors are necessary to divide it by 2. The reason for this is that we use the internal reference voltage of 2.56 V (this may be 1.1 V for other boards, so be warned!).
4. Pins 2 and 3 are used to drive the LCD. For Arduino Nano Every these should be pins **A4 and A5**.

Note that the LCD needs to have the I2C “backpack” soldered on. Here’s what it looks like:



Step 5: loading the Arduino sketch into the microcontroller

For this step you need to first download the Arduino code. You can find it under the GeigerDAQ repository, under the `./GeigerCounter` directory. We recommend that you clone the full repository. In addition to the Arduino sketch it also contains the python script, `logger.py`, which you can use to transfer data from the microcontroller to your computer.
To transfer the code to the device you need to install the latest version of Arduino software. You open the sketch GeigerCounter.ino, after which under **Tools->Port** you pick the port that connects to the microcontroller. You can then **Verify** and **Upload** your code (top left of the IDE).
Note: for newer chips you may have to install the latest libraries.

Step 6: putting it all together

Here we put all the components in the enclosure. But first you need to drill some holes:

- 4 holes to mount the LCD. See p2 to get an idea
- One (big) hole on the side of the enclosure for the micro-USB connection. Make sure to measure its position first – you want it to line up with the micro-USB connector

After this you need to

- use double sided sticky tape to glue the breadboard to the bottom of the enclosure (we suggest putting this as close to the hinge as possible) while aligning with the hole you drilled for the micro-USB connector. The breadboard should be right up flush against the hole.
- Use the same sticky tape to glue the battery pack to the internal side of the cover, in a way that you can later open it to change the batteries
- Use zip-ties to fasten the LCD to the cover

Refer to the pictures on p1 for reference. The position of the individual parts is very important in order for the LCD and the battery pack not to run into the components on the breadboard.

What does the Geiger Counter do?

It counts radiation hits, mostly gamma rays from various radioisotopes that surround us in our environment. Well, it does more than that. Below is a list of features of the fully assembled device.

The Arduino processes the signals from the Geiger tube and displays a variety of parameters:

- **Count rate** This rate is determined by a variable integration window. The lower the rate the longer the integration window. This feature is added to reduce the statistical fluctuations of the displayed result.
- **Dose rate** The count rate can be used to estimate the dose rate. This is done by simply multiplying the counts per minute (CPM) by 0.57 – the conversion to mR/hr. One should be warned that this is only true for the ^{137}Cs standard.
- **Data Transfer** Arduino can also be connected to the laptop and the data can be transferred over the serial bus. The transferred data essentially consists of the timestamps – in Arduino clock reference – of the events. This can allow for some interesting analysis later.
- **Battery voltage and HV** When powered the microcontroller measures two voltages: the “battery” voltage (or USB voltage, if that’s what the device is powered with), and the HV voltage. Both are displayed on the LCD for about 10-30 seconds at start time. To check those again simply turn the device off and on.
    - Idea 💡 : implement a button which will allow you to continuously display the real-time voltages. This would be quite useful when tuning the HV!