1 Introduction

In this two-week long lab, which will require a long lab write-up, you will build the data acquisition (DAQ) unit for a Geiger counter. You will compare the data you collect using your custom DAQ to compare with the theoretical expectations for Poisson and Gaussian distributed random variables.

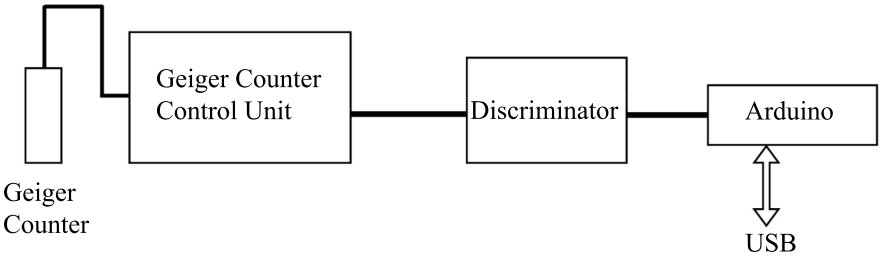


Figure 1: An overview of the lab setup. The Geiger counter and control unit are provided. You will build the discriminator stage and program an Arduino to complete the DAQ.

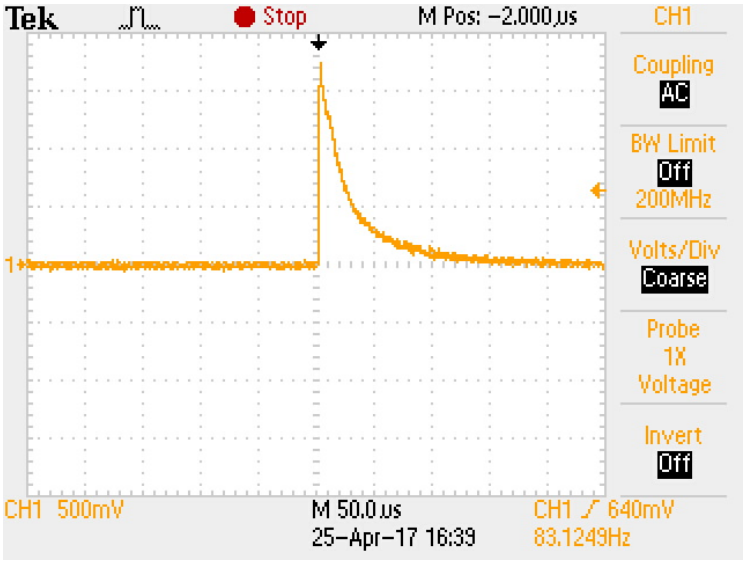


Figure 2: Typical pulse from the Geiger counter

An overview of the lab setup is shown in Fig. 1. Once properly setup, the Geiger counter will produce a pulse like the one shown in Fig. 2 each time ionizing radiation passes through the chamber. The scope trace is AC-coupled, and therefore does not show the 5 volt offset (called a pedestal) that is present at the output. The first stage of the discriminator removes this pedestal with a high-pass filter. The subsequent stages convert these pulses above threshold into 5 volt digitized TTL pulses. Once connected to the Arduino, the DAQ unit receives a single TTL digital pulse on an input pin for each incident ionizing particle. The DAQ simply counts these pulses during a fixed time interval and reports the results over the serial connection. You'll build the discriminator, the DAQ, collect the data, analyze it using scientific python, and report everything in a long writeup.

2 Precautions

Precautions with the Geiger counter:

- Leave the cable from the Geiger counter controller to the Geiger counter in place *at all times*. This carries voltages of approximately 1000 volts. If you leave the cable in place, nothing can be inadvertently plugged in (including fingers!)
- Leave the Geiger tube in its holder. It has a thin front window which is easily broken.

Precautions with the radioactive source:

- Don't touch the source.
- Leave the source in the tray at all times. The TA will provide the sources and handle moving them from place to place.
- Radiation falls off as $1/r^2$. So minimize your time near sources and maximize your distance from them.
- Further information on radiation safety can be found in Melissinos and Napolitano.

3 AC-coupled Schmitt Trigger

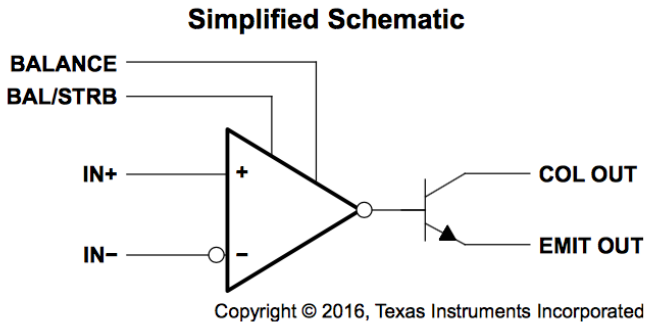
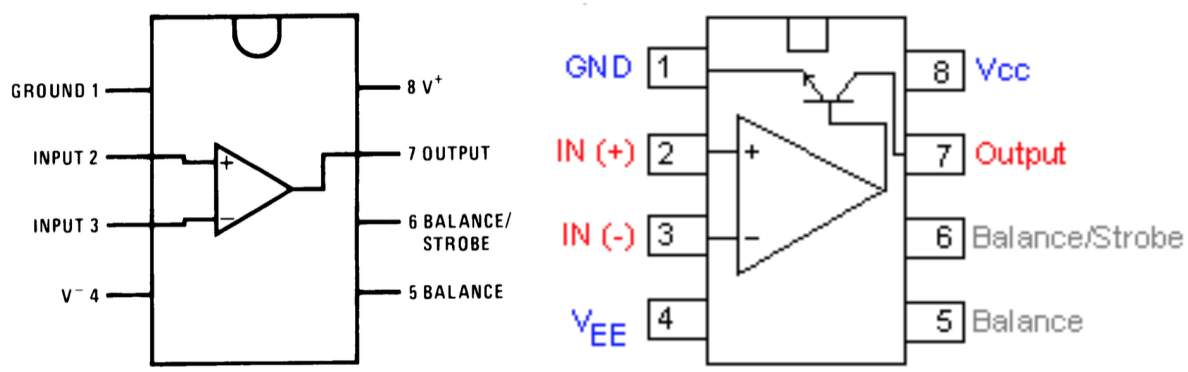


Figure 3: The LM311 schematic.

The first stage of the DAQ is a Schmitt trigger discriminator based on the LM311 that you will build yourself. The LM311 is intended for TTL (5 V) logic and can be driven with a single-ended voltage all the way down to 5 V. This makes it an ideal component to use with our 5 V Arduino Uno. We can power the LM311 from the Arduino 5 V supply and send it's output directly to the Arduino. There is no need to get out the bench-top DC supplies!

As shown in Fig. 3, the LM311 is an open-emitter open-collector comparator, that is, an op-amp driving the base of an NPN diode with both the collector and emitter available as outputs. Typically the emitter of the output transistor is tied to ground and the collector output is therefore either grounded or high-impedance. For this reason, a pull-up resistor is generally needed at the the collector output, so that the output is either V_{CC} or ground, depending on which of the two inputs is largest.



(a) (b)

Figure 4: The LM311 in 8-pin PDIP as (a) conventionally and somewhat deceptively shown, and (b) with the output transistor explicitly shown. The version (b) is still deceptive, as it leaves out the not operation between the op-amp and the transistor base, as shown in the device schematic.

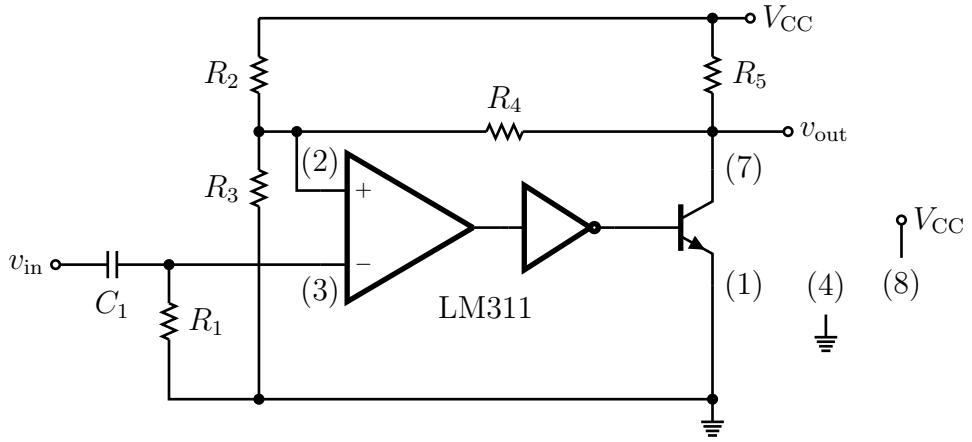


Figure 5: An AC-coupled inverting Schmitt trigger based on the LM311. The LM311 IC provides both the operational amplifier and the transistor. The numbers in parenthesis are pin numbers.

The discriminator circuit you will be building is shown in Fig. 5. The input to the discriminator is AC coupled by R_1 and C_1, which is needed to remove the pedestal (DC offset) present at the Geiger counter output. This input is compared to a threshold predominantly set by the voltage divider R_2 and R_3. When the input signal (at the *inverting* input) exceeds this threshold, the base of the transistor is driven high (note the NOT) and so the output is pulled to ground. When the input signal is below the threshold, the base of the transistor is driven low, and the output is at V_{CC}, due to pull-up resistor R_5. The feedback resistor R_4 add hysteresis to the circuit, providing stability by pulling the threshold slightly higher once the output is high, and slightly lower once the output is low.

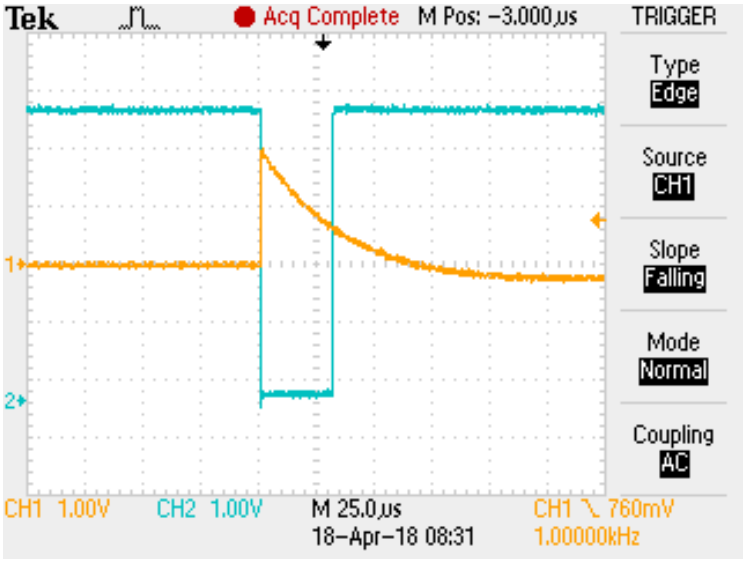


Figure 6: Typical output v_{out} of the discriminator circuit (at pin 7 of the LM311) compared to the input v_{in} (at pin3 of the LM311).

Build the circuit shown in Fig. **??** using $R_1 = 4.7$ kΩ, $C_1 = 10$ nF, $R_2 = 8.2$ kΩ, $R_3 = 1.1$ kΩ, $R_4 = 82$ kΩ, $R_5 = 1.1$ kΩ.

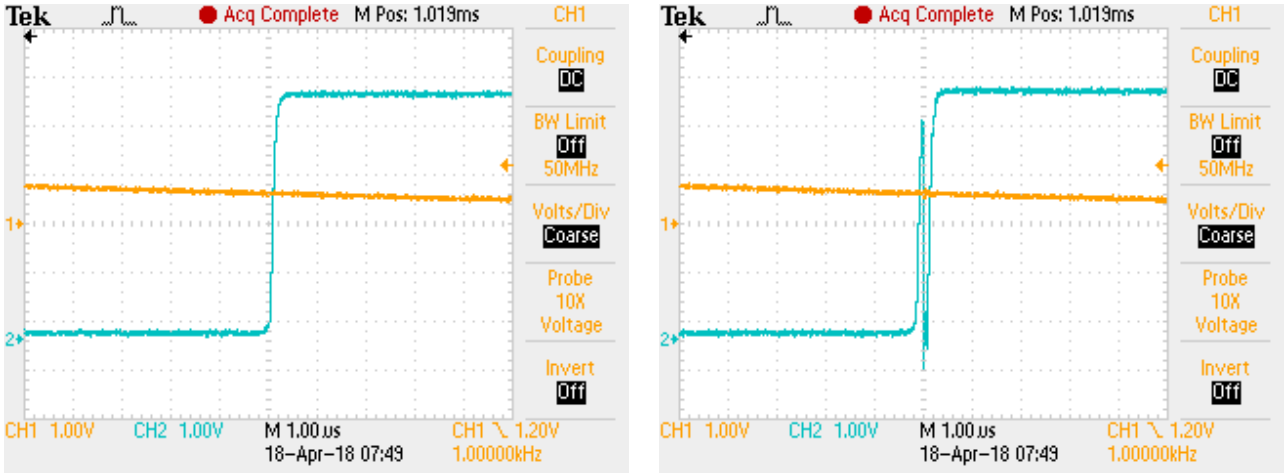
using the components in Figs. **??** and 3. For debugging, drive the circuit at V_{in} with a 1 kHz square-wave at 2 volts peak-to-peak. Typical output from the circuit is shown in Fig. 6. Note the different scale: the output is 5 volt TTL digital logic, easily handled by a microprocessor.

4 Arduino Based DAQ

The output of your discriminator circuit is easily handled by an Arduino microprocessor. The signal is a digital 5 volt TTL pulse with a width of order several tens of microseconds, which can be fed directly to a digital input pin on the Arduino. The Arduino should also be grounded to the same ground used in your discriminator circuit. Your Arduino, if programmed correctly, can make one digital read approximately every 5 microseconds, which is plenty fast enough to handle these TTL pulses.

You should therefore design a DAQ for your Arduino which:

- Measures the current time in microseconds.
- Makes a programmable number of iterations (N_I) reading the digital input each iteration.



(a) (b)

Figure 7: Single trace for the rising edge of the discriminator circuit (a) as designed, and (b) with feedback resistor R_4 removed from the circuit. Without feedback, there is no hysteresis, and the output is jumpy. Once you have your DAQ working, you can check what removing this resistor would do to the stability and accuracy of your rate measurement. Spoiler alert: you need it!

- Increments a counter (N_C) each time a new pulse is detected (a digital value "1" after a "0")
- Measures the stop time.
- Calculates the duration (τ) of the run, from stop time minus start time.
- On the serial bus, reports τ, N_I, and N_C.
- Repeats again from the start 100 times.

When designing your DAQ, consider the following:

- You might measure a "1" corresponding to single pulse multiple times. So the sequence 0001111110000 should increment your counter only once, not six times.
- We are pushing the limits of the Arduino timing, so you'll want to place the loop over N_I iterations directly in a for loop, which does the data collection, counting, and nothing else. Keep it simple! You should check that you manage to keep the sample rate τ/N_I at nearly 5 microseconds.
- For actually collecting the data, you can simply cut and paste the serial monitor output into a text file. It will be easiest to process the data later using scientific python if the output is simply one line per run, with the variables reported separated only by spaces. Of course you can add whatever output you want at the beginning and end.
- Verify that by adjusting the number of samples collected, you can control the interval τ to 100 ms and 1000 ms.
- Verify that your Arduino consistently measures the rate that you expect, based on the square wave input that you are feeding it, and that it does not depend on the value of τ.

Once you have verified that both your discriminator circuit and Arduino-based DAQ are working as expected, you are ready to start taking data.

5 The Geiger Counter

On the Geiger Controller, start with the HV set to zero at both the analog knob and the HV on-off switch. Turn on the device and set the mode to"Test". In this mode, the counter will be incremented at a fixed rate of 60 Hz. While in this mode, play around with the "Count", "Reset", and "Lab-Chron" analog timer reset dial until you understand how all of these features work.

Next check that you have a source in the second highest draw of the Geiger Counter holder, asking the TA for help if this is not the case. Switch the mode back to "Use", which will now use the Geiger counter as input to the count. With the HV still off, place the controller in "Count" mode, and verify that the rate, with no HV, is zero. Now, with the controller still in "Count" mode, turn on the HV and turn the dial until you first see the counter incrementing. Turn up the voltage to the next interval of 50 volts (e.g. if it first starts incrementing at 730 volts, set the dial to 750 volts).

Tabulate the number of counts in a 10 seconds interval, twice for each voltage level, in 50 volt steps from your starting voltage up to 1000 volts. Do not exceed 1000 volts. From the data, select the start of the plateau region, where increasing the voltage by 50 volts raises the rate by less than 10%.

With another measurement, confirm that the rate at this plateau is of order 100 Hz (this will vary from source to source).

Look at the "SCOPE" output of the Geiger counter controller on your oscilloscope, with AC coupling, and verify that you see output pulses similar to those in Fig. 2.

6 Connect the DAQ

Connect the "SCOPE" output of the Geiger counter controller to the input of your discriminator circuit. It is useful to use a BNC Tee connection at the scope so that you can continue to trigger and monitor Geiger counter pulses and drive your discriminator with the same input. Now check the output of your discriminator and verify that you have the expected TTL output pulse. If the width of the TTL output pulse is less than 10 microseconds, your Arduino may have trouble keeping up. If this is the case, seek help from the TA. Together, you can try increasing the HV (but do not exceed 1000 V) and (if that fails) you could try adjusting the capacitance of the high-pass filter stage.

With the shape of the digital input pulse verified, plug this in as input to your Arduino DAQ. Verify that the rate you obtain with the Aruduino is within about 20% of the rate you estimated with the timer and counter features.

7 Collect and Analyze your Data

Now you are ready to collect the data you will use for your analysis. From your measured rate, estimate the τ and therefore N_I needed to collect a mean N_C of 1,5,10. At each of these four values of τ, collect the count for 100 individual runs. Before leaving, verify that you the mean value for each run is near the target mean value! You could easily program your Arduino to take this average for you!

Using the scientific python techniques we will develop in class, compare the resultant distributions to the predictions from a Poisson distribution with a mean value λ matching that of your distribution.

Repeat this for a Gaussian distribution.

In your write-up, explain what you expect and what you have found.
If time permits, also collect data from 1000 individual runs at each value of τ.