PHYS 207 begins with cheap (terribly inefficient) SPADs

We have provided, in partnership with our professional societies, good-quality Single-Photon Avalanche Detectors (**SPADs**) along with hands-on methods for (so far) one-third of all U.S. Physics departments to teach about single photons, and quantum physics. This is a very big deal. Our goal is to develop a model program for the rest of the world to emulate. — The SPADs that we have distributed have high "Quantum Efficiency," but do cost about $ 1700 each, and can be accidentally destroyed through exposure to common levels of room lighting. So, in addition to implementing methods aimed at *protecting* these devices, we instead *start out* our PHYS 207 students with some terribly inefficient (but wonderfully cheap) detectors, so that they can learn some of the device principles without concern. — Some makes and models (but certainly not all) of common (*i.e.*, cheap) light-emitting diodes (LEDs) will, when reverse biased to very near "breakdown," act as a Geiger-mode avalanche detector. (Mostly, photons ***go right through*** these cheaper devices, without effect, but very rarely they will detect one.)

1) Students can use these reverse-biased LEDs to directly observe the current pulses generated by avalanche detectors, without having to deal with the high voltages associated with a Geiger tube.
2) As a precursor to creating a "counting circuit" (*i.e.*, a scalar) students first build a simple *comparator* circuit, which will **digitize** the pulse produced by the avalanche detector.
3) An inexpensive ("Teensy") microprocessor will then be used to complete a measurement system.

Along the way, students can:

- Learn more about the pulses associated with avalanche detectors, including the quenching of pulses, the kinds of time constants associated with such devices, and "dead time" (and how to measure dead time using time-between-events distributions), and "after-pulsing" in avalanche photodiodes.
- Learn something about semiconductor device physics, including the temperature dependence and spectral response of the LED/SPAD.
- Learn to use electrical "breadboards," power supplies, and oscilloscopes
- Learn something about operational amplifiers
- Learn about comparators (which helps with understanding the triggering of oscilloscopes)
- Learn common methods of *signal conditioning*, as a precursor to further electronic processing
- Learn common "rules of thumb" used for histogram plots
- Use a single-photon detector to illustrate properties of random counting experiments
- Use limiting probability distributions to perform statistical analysis on a physical system.

Required reading from Taylor, *An Introduction to Error Analysis*

- Histograms and the *Normal* Distribution (Taylor 5.1-5.3: pp. 121 - 135)
- A **homework problem** on the *Exponential* Distribution (Taylor Problem # 5.6, p. 155)
- The *Poisson* distribution (Taylor Ch. 11: pp. 245 - 254)
- Summary of the Square-Root Rule for a Counting Experiment (Taylor 3.2: pp. 48 - 49)

Your **lab notebook** should contain descriptions (including references) of: p-n junctions, LEDs, photodiodes, and avalanche photodiodes: explain the physics behind the behavior you expect to observe. Some references to consider might be:

[1] E. Rutherford, *Radioactive Substances and Their Radiations* (Cambridge University Press, 1913).

[2] D. Renker, "Geiger-mode avalanche photodiodes, history, properties and problems," *Nuclear Instruments and Methods in Physics Research A*, **567**, 48-56 (2006).

[3] S.M. Sze, *Physics of Semiconductor Devices*, 2nd Ed. (Wiley, 1981).

[4] M. Levinshtein, J. Kostamovaara, S. Vainshtein, *Breakdown phenomena in semiconductors and semiconductor devices* (World Scientific, 2005).

[5] E. Hergert, S. Pitek, "Understanding key parameters of silicon photomultipliers," *Laser Focus World*, November 2014, pp. 45-49 (www.laserfocusworld.com)

[6] P. Horowitz & W. Hill, *The Art of Electronics* (Cambridge University Press, 1989.

Single-Photon Detectors (a *long series* of labs)

If I were a geeky young student, rather than a geeky old professor, phrases that would excite me *might* include "Quantum Information," or "Spy Technology" or "Next-generation Internet" or "Programmable Optics for Virtual Reality, Augmented Reality, and Mixed Reality" or "Autonomous Vehicles" or "Machine Vision." It turns out that the opportunities in Optics & Photonics extend well beyond those areas, but generating student excitement has been a key point of discussion recently. Accordingly, the Physics department has voted to implement a new **CONCENTRATION** in Optics & Photonics (which could, alternatively, be a **MINOR** for those not majoring in Physics). In part, this reflects a common interest among the faculty members of the department, but also reflects an area of significant opportunity for our students. Those who go on to get, say, a Masters in Optics & Photonics will command significantly higher starting salaries than nearly any other kind of Engineering (see *Optics & Photonics Global Salary Report*). Of course, going on for a Masters degree is just one out of four distinct paths our students take to become Engineers. We also offer, for example, the 3:2 dual-degree (Physics + Engineering) option, where it turns out that our "Mathematical Methods in the Physical Sciences" course (**PHYS 304**) is a structural "backbone," giving our students what typically turns out to be a stronger *applied* math background than the average student who went directly to an Engineering school as an incoming first-year student, and is now entering their third year of university. This stronger *applied* math background is one of several reasons our 3:2 students typically have higher-than-average GPAs at our partner institutions, and we have been looking at ways to *increase* the advantage our students gain, informed by ***co-valuing*** **the formalisms of physics with the hands-on experimental "grapplings," and with the (computer-based) programmatic approaches**. — By highlighting Optics & Photonics in ways that reinforce this "three-legged stool" of student development, we add significant marketability to their portfolios. Our goal is to create a very high-quality **series of scaffolded courses**, which feed into a strong system of support and engagement, up through student participation in a "Signature Work Seminar," which our students will take in their final semester at IWU, extending projects that of interest to them. — Whereas PHYS 106 only asks students to map out optical systems by considering a few "sample" rays of light, PHYS 307 ("Optical Physics") and "Scientific Imaging" (PHYS 308) show that more powerful treatments consider the superposition of "*many* paths," and provides computational tools to promote that kind of power. PHYS 207 complements these, and stands alongside Quantum Optics: The Momentum of the Photon (PHYS 317), which provides further introduction to single-photon quantum mechanics, offered in parallel to our traditional upper-level E&M (PHYS 406) and Quantum (PHYS 407) courses. Our approach is unique in that it presents Quantum Optics (a formalism of wave-particle duality applied to light, in PHYS 207 and 317 and 407) alongside *parallel* discussion of Classical Electrodynamics (a formalism appropriate in the limit of astronomical numbers of photons, in PHYS 106, 307, 308, and 406), presented as a story of energy and momentum. Our aim is to leverage clear physical understanding provided by analysis of laser beams (containing astronomical numbers of photons), so as to provide context for discussions of some of the "weirdness" of modern physics, which emerges quite strikingly at the ***single***-photon level.

Procedures adapted from Jonathan Newport, American University:

Exercise #1: Lighting an LED ("forward bias")

A Light-Emitting Diode (LED) is a *non*-linear circuit element that can produce a controlled amount of light. The current flowing through the diode is *not* linearly proportional to the voltage across the diode, and yet in spite of that nonlinearity, model AND114R provides luminous intensity that is linearly proportional to the current flowing through the LED.

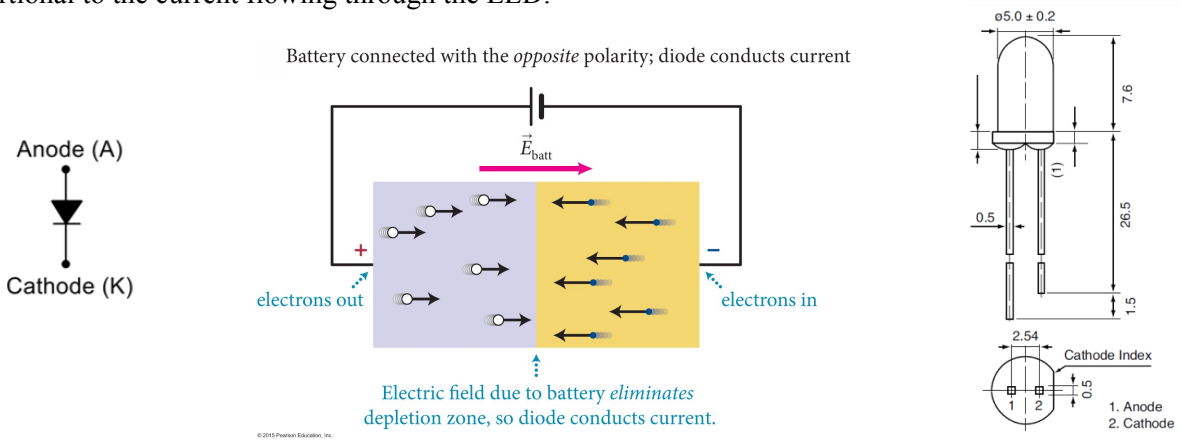
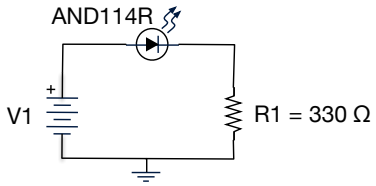


Fig. 1. Forward-biased PN junction diode schematic symbol (left), physical cartoon (center) [after Mazur, Principles & Practice of Physics], and **pinout diagram** (right)

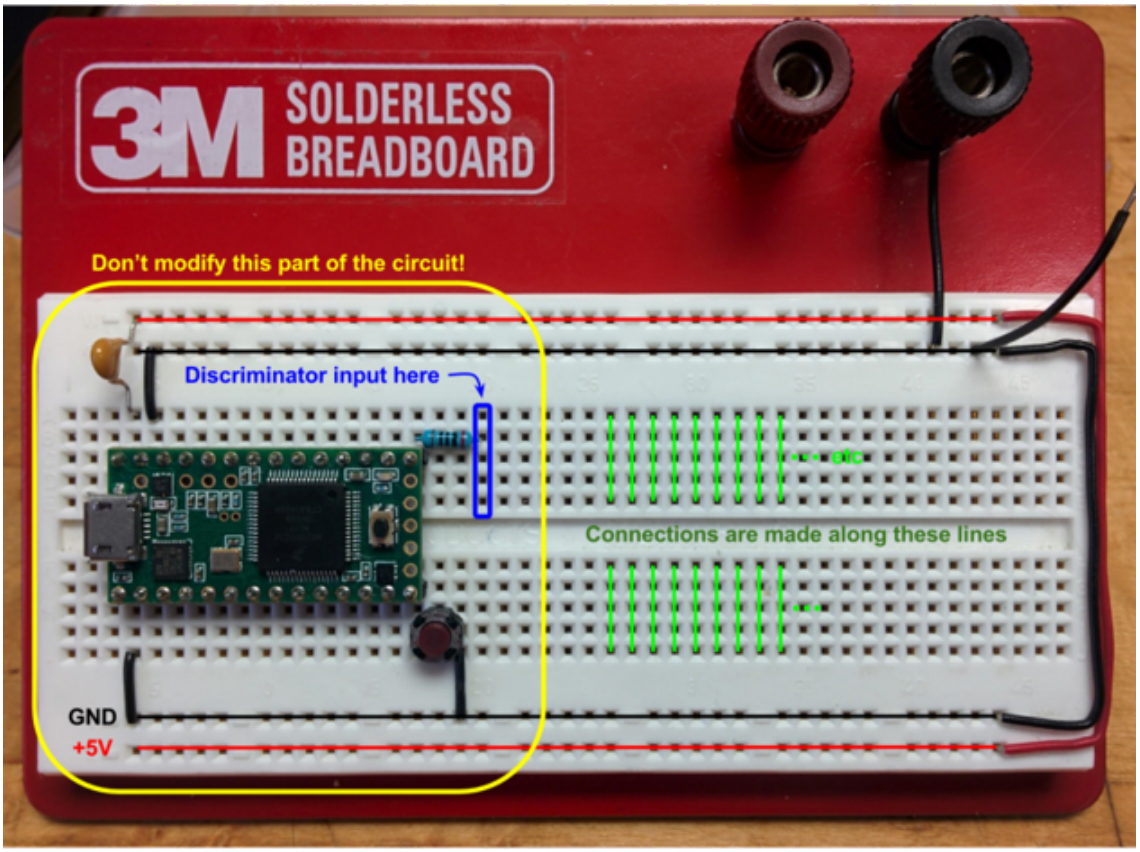
Diodes behave like a one-way valve for current. When the voltage applied to the *anode* is more positive than the voltage on the *cathode*, then the diode is said to be in *Forward Bias*. Under these conditions, electrons from one side of the PN junction, and holes from the other side, flow into the depletion zone, and recombine, a process which results in the emission of light.



When forward biased, as the voltage across the diode increases, the current through the diode increases *dramatically*. The heat generated by this current can easily destroy the device. It is always wise to install a current-limiting resistor *in series* with the diode (as shown above) to prevent thermal runaway.

Procedure:

1. Before starting, adjust your power supply, V1, so that the voltage control is near 0 Vdc, and so that you can **monitor the current** it outputs with the *built-in* ammeter (***NOT*** an external ammeter!)
2. Construct the circuit shown above, using the proto-board shown on the next page.



Using an electrical breadboard

A *solderless* breadboard is used, here, to make connections between components and provide connections to external power. Individual rows at the top and bottom of the breadboard are electrically connected, and are generally reserved for power connections. In the photo above, the inner rows denoted with black lines are used for zero potential (a.k.a. "ground" or "0V") and the outer rows denoted with red lines are used for positive voltage supplied by the Teensy (in this case +3.3V). — In the central portion of a breadboard, individual columns of **five** insertion points are electrically connected (think of them as the five fingers of a single "hand"). These are denoted with green lines in the diagram above. The central "trough" is useful to separate connections on standard integrated chips.

3. Using a Digital Multimeter (DMM), measure and ***record*** the voltage drop just across the LED.
4. Slowly increase the voltage control until the LED begins to produce light, making sure that you. ***DO NOT EXCEED 20 mA of current***. [As always, **plot your data *as you take it!***]
5. Does your "voltage drop" across the diode at 20 mA *agree* with the values found by other students? (To what degree is it "*invariant*" across different devices?)

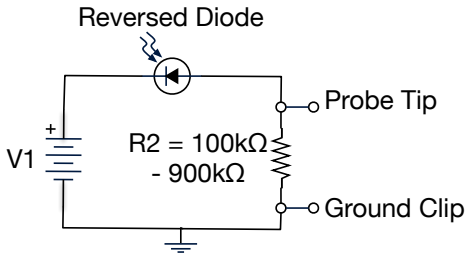
Exercise #2: Detecting Photons ("reverse bias")

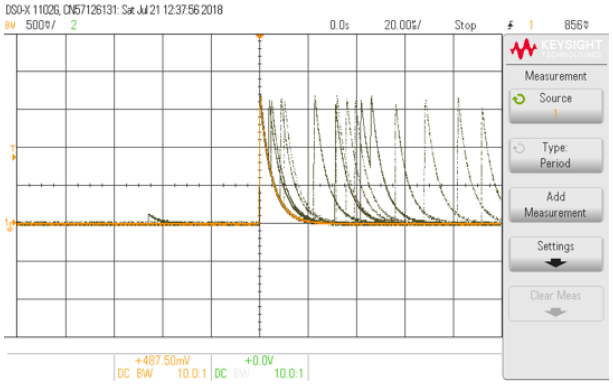
LEDs, as their name suggests, are intended to be used as light-*emitting* devices. However, the semiconductor physics that governs their behavior also allows them to be used as light *detectors*.

When the voltage on the Cathode is more positive than the voltage on the Anode, the diode is said to be in *Reverse Bias*. For typical reverse-bias voltages, very little current flows through the device (*essentially* zero current). Even when the reverse bias is small (nowhere near *breakdown*), LEDs can be used as photodiodes (essentially tiny solar cells) to detect light near their own wavelength, but under those operating conditions the response of the LED is typically linearly proportional to the intensity of the incident light, and the LED will not be capable of detecting ***single*** photons of light.

When the reverse bias voltage is *large enough*, the diode will start to conduct on its own (even in the dark) – this process is called *breakdown*. For many diodes, the onset of breakdown happens for voltages of order 1000V, and the effect can be **EXPLOSIVE!** However, for the particular LEDs that we will use, the breakdown voltage is only around 25V and (critically) the capacitance of the junction is so little that the amount of energy stored will not cause explosive damage, and so they (usually) survive. That is, if you ***carefully approach*** an operating point *near* breakdown, you can detect ***single*** photons of light, exploiting the fact that the system is unstable enough that the input of even a solitary photon can (sometimes) result in momentary pulse of current flowing through the reverse-biased LED. A single photon can give rise to cascading "breakdown" avalanche; by then passing this current pulse through a large resistor, a measurable voltage signal is produced, read by an oscilloscope.

Many of the single-photon detectors used for cutting-edge projects, such as the Photomultiplier Tubes (PMTs), Silicon Photomultipliers (SiPMs), superconducting bolometers, and specially optimized avalanche photodiodes (APDs) can be rather expensive, can (**like the Geiger tube**) require dangerous high voltages, and can also be easily damaged. Fortunately, a few intrepid physicists found out that there are a handful of inexpensive LEDs that can survive the same kind of *avalanche breakdown* required of ***single***-photon detectors. These LEDs are *not* designed for this purpose, so they make *rather poor* single-photon detectors, but they are useful for the didactic exercises that follow. In this experiment, one of these special LEDs is reverse biased and as the bias voltage is increased, the LED becomes a (***low efficiency***) avalanche photodiode.





Procedure:

1. Begin with your voltage source V1 adjusted so that it is near 0Vdc.
2. Noting that the LED **should now be in reverse bias**, construct the circuit above (**R2 >> R1**).

3. Use an oscilloscope probe to monitor the voltage *across the resistor*.

a. Make sure that both your oscilloscope probe and oscilloscope channel are set to the ×10 setting. (Triple check!!!).

b. Start with the oscilloscope vertical scale on 100mV/div and horizontal scale on 10μs/div.

c. Trigger on the active channel and set the trigger level to ≈100mV.

d. When I was being mentored by Prof. Jonathan Newport, I was able to try these labs using a model AND113R LED, which starts to act like an avalanche photodiode at ≈ 26 ± 2V of reverse bias. What I ordered for our lab is a similar model, the AND114R, so I'm guessing that the breakdown voltage. is likely to be similar. Approach this value slowly until you start to see pulses. Make small adjustments to convince yourself that you are not merely in the "self-ionization" limit, but that the pulses you observe are very likely being triggered by photons coming from outside the device. We *claim* that these pulses represent ***single*** photons of light striking your detector! (It is only much later that you will be in position to experimentally determine whether or not these pulses each correspond to *single* photons.) For now, what can you do to confirm that this circuit is indeed acting as a photodetector? — **RECORD YOUR OPERATING VOLTAGE!** (I look forward to *learning* what voltage level worked for you!)

e. Adjust circuit and oscilloscope parameters and make observations, including:

- Reverse-bias voltage level
- Trigger level
- Current-limiting resistor values (keeping them between 100kΩ-900kΩ). Why does the pulse ***shape*** change with different resistors? Reminder: the LED has a *junction capacitance*. Describe the pulse shape, in words, in your notebook. How would you **quantify** the time constants?

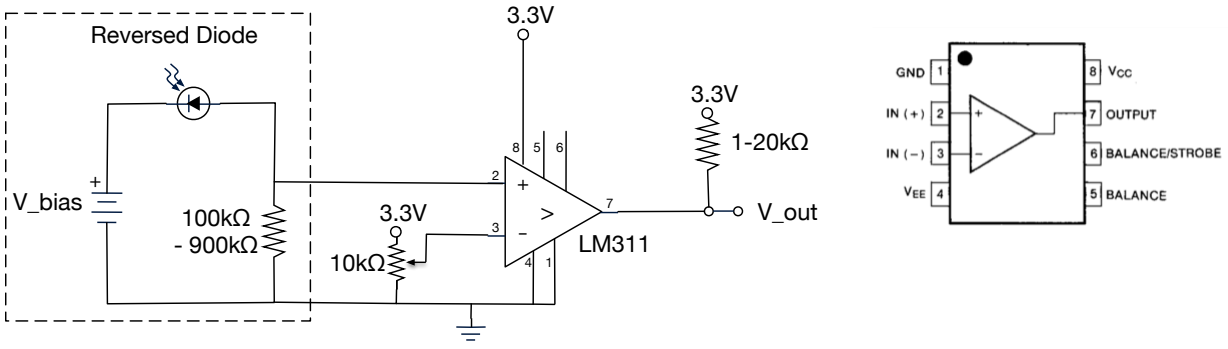
f. Use *LabVIEW* to capture some of your data and include this, with analysis, in your laboratory notebook. [Caution: whenever faced with a model predicting an exponential decay, it can be helpful to switch back and forth between a linear plot and a semi-log plot. *Each can provide important insights!* Also, please be reminded, as seen in your data using aluminum absorbers, that simple models typically have a limited range of validity, and that data outside that range can reasonably be excluded from any fit or application of that model, so long as you clearly communicate, to your audience, what you have done.]

Exercise #3: Signal Conditioning ("digitization")

Higher-quality versions of the Single-Photon Avalanche Diode (SPAD) detectors you're now working with now find commercial application in *quantum key distribution* (currently in use in financial markets as well as applications relating to national security), *laser ranging* (in mass market applications such as focusing the camera in your cell phone), *florescence microscopy* (which is absolutely essential for biomedical work), *etc*. Each of these systems consist of a detector integrated into a "*measurement chain*," which we envision as a modular string of discrete stages, where *information* is passed (typically in the form of a voltage) from one stage to the next.

In Exercise #2, you constructed a "first stage" which converted a *current* pulse, generated in your SPAD detector, into a *voltage* pulse (simply by passing the current through a resistor). This week you'll add a "second stage" to that, which you will use to perform *signal conditioning* that is helpful before you add a "third stage" to your measurement chain (in a future exercise). In the schematic below, note that the *dotted* portion is just what you'd *already* put together in Exercise #2, so your first step will just be to *reconstruct* that, if need be, and to **make sure that it is working correctly**. As before, make sure that *both* your oscilloscope probe and oscilloscope channel are on the ×10 setting. You will be keeping that first scope probe in place even after you add another modular stage to your system.

To perform counting experiments as well as precision **timing** measurements, using pulses coming from an avalanche photodetector, you'll next convert the (still *low-level*) raw pulses coming from your detector into a standard (*high*-level) digital signal that can be unambiguously interpreted by a digital readout system. The circuit to convert the "raw" pulses from your circuit in Exercise #2 (shown inside the dashed square below) into *digital* signals is called a ***discriminator***, a term that originated in the High-Energy Physics community (where a significant portion of all methods for high-speed signal processing were first developed), and describes a circuit that provides an output signal only when an input signal exceeds some threshold value (a "trigger"). You've used such circuitry previously, for example when you were adjusting the trigger level on the oscilloscope in Exercise #2, you should have noticed that more pulses were displayed on screen when the trigger level is low.



Among the additions you'll make today, the key component used to effect a discriminator is called a ***comparator*** (shown in the circuit above as model LM311, with the chip *pinout* shown at right). Essentially, a comparator is just an operational amplifier (*op-amp*), but without the negative (stabilizing) feedback that would normally redirect some of the output back into one of the inputs. In order to have some sense of what, physically, happens inside an op-amp you'll need to get far enough in your review of

the physics of the p-n junction, and of transistors. Operationally, whenever the voltage presented at the non-inverting input (labeled with a "+") of the comparator rises to a level *greater than* the voltage at the reference input (labeled with "-"), then the output voltage will rapidly swing all the way up to +3.3Vdc, where anything above +3Vdc corresponds to a "1" in newer digital logic (*previously* common digital logic treated +5Vdc as a "1" but as we pack more sophistication into our chips, the spacing between electrodes is reduced, and newer devices often cannot tolerate the older logic levels). Conversely, whenever the voltage at the non-inverting input (+) of the comparator drops below the voltage at the reference input (-), the output voltage will rapidly pop back down to 0Vdc. (Digital logic now typically treats anything below +1.5Vdc as "0," but none of your input pulses are that large, so you require an *adjustable* trigger level.)

For the moment, you'll have to take my word on the two **Golden Rules** of an *ideal* op-amp:

1. The output changes in a way that would, if feedback were in place, try to eliminate any *difference* between the voltage of the two inputs.
2. Essentially no current flows in or out of the inputs.

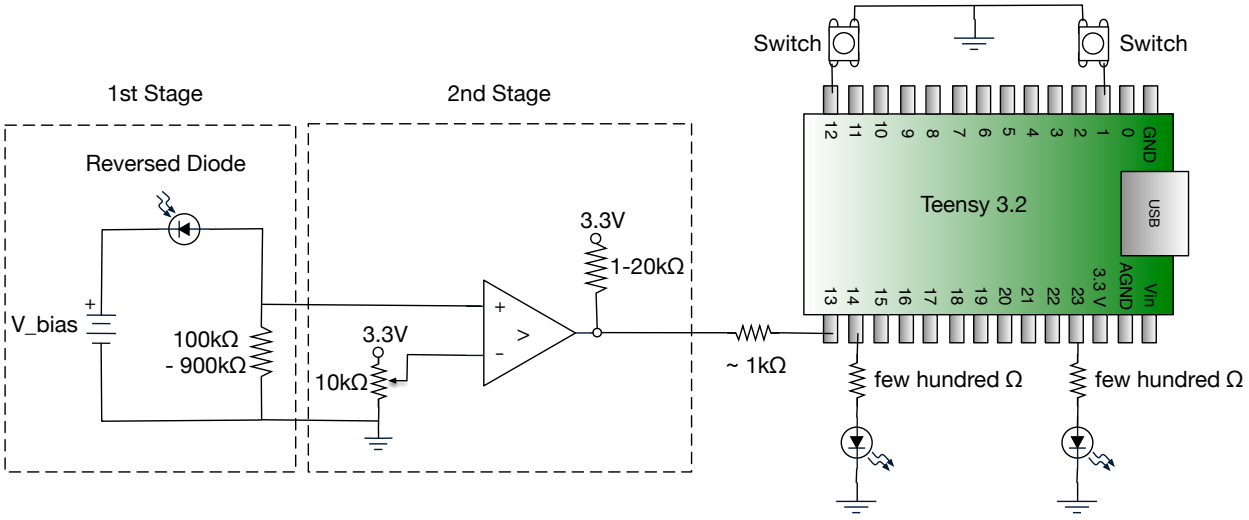
In your notebook, explain how the new part of the circuit above will allow you to turn your (initially *small*) **voltage pulse into a pulse with an amplitude of ~ 3 V.** [*Hint: invoke the Golden Rules!*]

Procedure:

a. Leave your circuit *and scope probe* from Experiment #2 in place and, after examining the photo above, add in the extra elements required for your discriminator. Because it will be used for later stages in your measurement chain, we elect to provide +3.3Vdc power to your circuit using the onboard power supply of the Teensy microcontroller: you need to connect the micro-USB connector to a computer! On the other hand, V_bias for your photodetector should be set using the same external power supply as before, to the same **OPERATING VOLTAGE** you recorded in your results from Experiment #2. Do **not** connect that large voltage to the horizontal rows!
b. By adding a second scope probe, you can *simultaneously* display the photodetector's pulses and the output of the discriminator on an oscilloscope. Adjust the "discriminator trigger level" (set by the variable resistor) so that the output provides one conditioned pulse for every photon pulse.
c. Measure and record the discriminator trigger level. [You may need to adjust this later on, to mitigate the number "false positives" or "dark counts" in your measurements (see Appendix C).]
d. Use *LabVIEW* to record at least one oscilloscope trace illustrative of your working discriminator.

Exercise #4: Counting Statistics (redux)

This experiment explores the ***statistics*** of counting experiments. Recall that, in performing their breakthrough work on radioactivity, Rutherford and his collaborators[1] developed methods for detecting, and counting, single particles emitted from radioactive substances. You know know that, today, we have a strong interest in the statistics of various kinds of *light* sources, and whether (or not) they are indeed random (and whether or not they exhibit "photon bunching"); not only are the following tests analogous to Rutherford's historic experiments with radioactivity, they prepare you for research in **Photonics**. *For this, read about the theory of the binomial, Poisson, Gaussian (normal), and Exponential distributions*.



Procedure:

1. Although not all shown in the schematic above, all of the connections that you made last week, in the 2nd stage, should remain in place! [It is convention to not show "the rails" in schematics but, as you are now starting to realize, op-amp circuits will not work without them!] After ensuring that everything from last week is still working, **disconnect** the USB and **turn off** the V_bias applied to your SPAD.

2. Connect the output of the 2nd stage of your measurement chain (your discriminator) to the microcontroller's input: you do this by adding a 1K resistor that connects pin 7 of the comparator to pin 13 of the Teensy microcontroller. (You could have used a frequency counter to record data by hand, but the automated nature of a microcontroller data collection process makes things *much* easier.)

3. Connect a resistor (a few hundred ohms) to pin 14 of the Teensy, and then add, *in series*, a diode, which then connects to ground. This diode should be forward biased.
4. Repeat Step (c), with pin 23 of the Teensy. This diode should be forward biased.

5. Add a **button switch** that can toggle a connection between ground and pin 12 of the Teensy.
6. Add a **button switch** that can toggle a connection between ground and pin 1 of the Teensy.

7. Re-apply the V_bias applied to your SPAD, and re-connect the USB to the Teensy. Last week's signals should still be there.

A. If using an IWU computer, open the Arduino application. If using your own laptop, click the following link to prepare your system.

B. *Load*, via the "right arrow" icon at the upper left of the Arduino app, Henry's code: SPADCounter02.ino — *Skim* the code, noting that there are two modes of operation: you'll first use ***FREQUENCY*** *counting mode* instead of the ***INTERVAL*** counting mode, which will be used in Exercise #5. Note: an ***integration time*** must be specified. (This is your count interval, which you had often set to 30 seconds for the Geiger tube, but occasionally set to 10 min. Here, the default is 1000ms.)

C. In the Arduino application, click VERIFY. Ask for help, as needed. Once set, you can open the *Serial Monitor* (the upper right button on the Arduino app) to communicate with the Teensy.

D. Open an Excel spreadsheet and CLICK on a cell where you would like data entry to begin. Then , on the Teensy microcontroller, press the WHITE button, and *then* click on the particular **button switch** that can toggle a connection between ground and **pin 12** of the Teensy. DO NOT touch your computer until you **collect photon frequency data over *at least* 100 integration periods**. What environmental variables need to be kept constant for a valid dataset? Your frequency data points should all be between 1000-5000 (which may require you to *adjust* the V_bias applied to your SPAD and re-start your data run). To *stop* your data run, click the *button switch* that began your run. [If you have time, you might collect photon frequency data over *a couple thousand* integration periods: this won't change the standard deviation, but will yield a better approximation to the limiting distribution, improving the standard deviation *of the mean,* $\sigma_m = \sigma/\sqrt{N}$.] Import a valid dataset into *Igor Pro*. **Correct the data for the *background* signal**, if one exists. [Use Chauvenet's criteria, as appropriate, to remove data points.] Ensure thorough *documentation* of your methods.

Statistical Analysis:

For the entire *valid* dataset, calculate the mean $\bar{x}$, standard deviation σ, variance σ^2, and the standard deviation *of the mean*. An explanation of what σ_m means, and what it is used for, is *required*.

- Next, calculate the average absolute deviation $d' \equiv \left\langle \left| x - \bar{x} \right| \right\rangle$ for the whole *valid* set of readings. *If* the data is Gaussian, you should be able to show (*i.e.*, derive) that the average absolute deviation should tend toward

$$d' = \sigma\sqrt{\frac{2}{\pi}}$$

  Calculate d' for your data set and see if it agrees with this expected relationship.
- Prepare a **histogram** of the data, complete with error bars. — As a "rule of thumb," choose your number of bins so that the highest bin has *something like* a fifth of your data. Both for dealing with histograms and for understanding counting experiments, please review:
  - Histograms and the Normal Distribution (Taylor 5.1-5.3, pp. 121 - 135)
    - The **required problem** on the *Exponential* Distribution (Taylor Problem # 5.6, p. 155)
  - The Poisson distribution (Taylor Ch. 11, pp. 245 - 254)
  - The Square-Root Rule for a Counting Experiment (pp. 48 - 49)
- You may compare your data to Gaussian or Poisson distributions, using the mean and σ from your data to create theoretical distributions of these forms. *Discuss the relative merits of each* of these for describing the collected data. A χ^2 test can provide an estimate of the goodness-of-fit between theory and experiment, but actually doing fits of your data to these functions is *extra*.
- Report the mean value of the frequency data and its associated uncertainty with the correct number of significant figures. Compare σ to the mean.

Exercise #5: Time Distribution of Counts – Temporal *Interval* Analysis

Hans Geiger and Ernest Marsden collected experimental data on radioactive materials at the University of Manchester in the early twentieth century. Under the direction of Ernest Rutherford, their experiments provided the first experimental evidence of the nucleus. Their data collection methods were similar to those that will be explored in this experiment – namely, taking the elapsed times between radioactive events as observed on a phosphorescent scintillator. Geiger and Marsden measured time intervals with a stopwatch. The rates we deal with in this experiment are much higher, so we seek an accurate and fast electronic means of data collection.

i. For this portion of the experiment you *must* use the functionality of the microcontroller – no other instrument in the lab is capable of autonomously collecting timing data between events!).

ii. Choose a *duration* for your experiment (the default is 1000ms = 1 second), in Henry's code, which you'll be using *in **INTERVAL** counting mode* instead of the ***FREQUENCY*** counting mode, through the use of a different *button switch*, as noted in the next step.

iii. Open the Serial Monitor and an Excel spreadsheet. Click in the cell where you want data entry to begin, *then* click on the particular **button switch** that can toggle a connection between ground and **pin 1** of the Teensy to commence data collection. The screen should populate with time-between-counts, reported in *micro*seconds, over the course of ten seconds. To *stop* your data run, click the *button switch* that began your run. Import this data (*tip*: use CTRL-A to select all data) into your favorite data analysis application.

Statistical Analysis:

- For a valid dataset, create a histogram, *complete with error bars*. Change the plot to a ln-linear scale. It is best to display this data as a *scatterplot*, instead of the default *bar chart*.
- *Where* data is truly random and exhibit *Poissonian* statistics, the limiting distribution should be *exponential*. Linearize your data and, where appropriate, find the slope and, critically, the y-intercept. (What do these *mean*? See Taylor Problem 5.6!!)
- Given an *adequate* fit, the mean can be calculated in three different ways: via statistics, the y-intercept, and the slope of your exponential fit. Discuss the relative merits of each.
- Report the mean and its associated uncertainty with the correct number of significant figures.

Additional Exercises:

The parameter space to *explore* in this lab is extensive. Some suggestions:

a. Extend your previous data sets to ***think further about the "dead time" of your detection circuit***. (Earlier, by plotting, the pulse "decay" as a function of resistance, you should have experimentally determined the capacitance of the photodetector via $\tau = RC$, for a given V_bias.)

b. Measure the "*dark count rate*" of pulses, by completely covering the photodetector. This data can be used to subtract a "***background***" from your data (see Taylor Section 11.4).

c. Create a constant source of illumination using an LED. Measure the count rate as a function of LED Current and/or luminous intensity.

d. Estimate the *quantum efficiency* of the photodetector.

e. Investigate the effect of *temperature* on count rate.

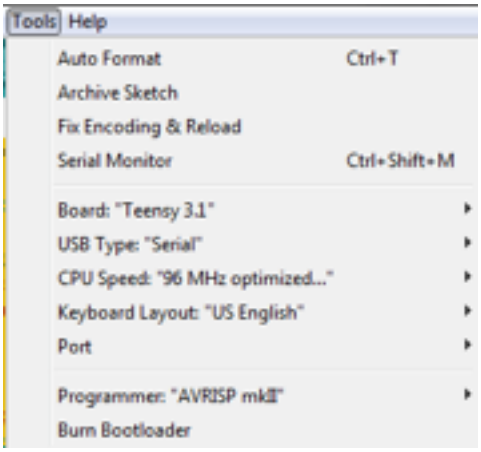
f. Investigate the *spectral response* of the photodetector.

Appendix A: Programming the Teensy Microcontroller

The Teensy microcontroller connects to a computer via a micro-USB cable. Your instructor has already installed the Arduino software it utilizes:
https://www.arduino.cc/en/Main/Software

Your instructor has also installed the Teensyduino software (with all libraries installed – we will make particular use of the FreqCount library):
https://www.pjrc.com/teensy/td_download.html

Connect the microcontroller and open the Arduino.exe program. The following settings need to be selected in the Tools menu for this to function properly:



To program the Teensy, open Henry's version of the code you'll need (or, alternatively, copy and paste the code shown on the next page) and click the **Upload** button as shown above. Open the **Serial Monitor** to communicate with the Teensy over the serial interface.

```
//-------------------------SPADCounter02.ino-------------------------
#include <FreqCount.h>//frequency counting library
#include <Bounce.h>//debouncing library

const int sensePin = 13;//pin for detection events
const int intervalPin = 1;//pin that sets the system into interval counting mode
const int freqPin = 12;//pin that sets the system into frequency counting mode
const int intervalLED = 23;//status indicator for interval mode
const int freqLED = 14;//status indicator for frequency mode

byte oldState;
volatile unsigned int bangTime = 0;//variable created to store the number of
microseconds between detection events
volatile int dataSent;//logical value of whether or not the NEW time interval data has
been sent. 1 true,0 false
const unsigned long senseLength = 1000;//time DURATION to collect interval data
(specified in milliseconds)

Bounce intervalButton = Bounce(intervalPin, 25);//bounce is a library used to
debounce. Bounce is the appearance of spikes of electric signal during
//logic signal switching, such as mechanical effects. Bounce(uint8_t pin, unsigned
long interval_millis ) creates an instance of the
//Bounce class, attaches pin and sets interval to interval_millis. The effect is to
remove bounce of the pin with the chosen parameter. See github bounce libraray
//for more info
Bounce freqButton = Bounce(freqPin, 25);//freqButton is a bounce instance, and the
instance is attached to pin 12 (freqPin=12) with interval 25.
//Bounce is used for mode switching pins because they are attached to mechanical
buttons which need debounce to be stable.

void setup() {
  pinMode(1, INPUT_PULLUP);//set mode of pin to resistive pullup input with internal
pullup resistor. Default electrical value would be HIGH, and becomes LOW when
  //a grounded button is pressed. Search pullup resistor for more information.
  pinMode(12, INPUT_PULLUP);

  pinMode(intervalLED, OUTPUT);
  pinMode(freqLED, OUTPUT);
  digitalWrite(intervalLED, LOW);
  digitalWrite(freqLED, LOW);

  Serial.begin(115200);//set baud rate of serial communication

}

int countMode = 0;     //mode = 0: nothing happens; mode = 1: frequency counter
operational; mode = 2: Interval Timer
elapsedMillis elapsedTime;
elapsedMicros senseTest;//a variable that increases as time goes to store the time
elapsed between events
```

```
void loop() {
//pins assigned to a bounce instance can be called (read change in value, read value,
detecting edge, etc.) with bounce methods.
//update, fallingEdge are some bounce methods. See bounce github for more info
  if (freqButton.update()) {
    //if the frequency mode button updates
    if (freqButton.fallingEdge() && countMode == 0) {
//only triggers the code to start the frequency counter when the button is pressed,
which corresponds to LOW electric signal. Remember the pin is in resistive pullup
mode.
//fallingEdge is used so that risingEdge does not trigger the code again to avoid
double triggering.
//This starts frequency counter when the button is pressed and when we are in nothing
mode.
      FreqCount.begin(1000);//FreqCount is a library for counting the number of events
over an INTEGRATION TIME (specified in milliseconds; default setting = 1000)
      elapsedTime = 0;
      Keyboard.print("time [ms]");
      Keyboard.set_key1(KEY_TAB);
      Keyboard.send_now();
      Keyboard.set_key1(0);
      Keyboard.send_now();
      Keyboard.println("Frequency [Hz]");
      digitalWrite(freqLED, HIGH);//indicate frequency counting is in progress
      countMode = 1;//show that we are in frequency counting mode.
    }
    else if (freqButton.fallingEdge() && countMode == 1) {
      //if we are already in frequency counting mode and the button is pressed, stop
the counter.
      FreqCount.end();
      digitalWrite(freqLED, LOW);//indicate frequency counting mode is over
      countMode = 0;//show that we are back to nothing mode
      Keyboard.set_modifier(MODIFIERKEY_CTRL);
      Keyboard.set_key1(KEY_HOME);
      Keyboard.send_now();
      Keyboard.set_modifier(0);
      Keyboard.set_key1(0);
      Keyboard.send_now();

    }
  }

  if (intervalButton.update()) {
    if (intervalButton.fallingEdge() && countMode == 0) {
      digitalWrite(intervalLED, HIGH);
      countMode = 2;

      Keyboard.print("integration time [ms]: ");
      Keyboard.println(senseLength);
      Keyboard.println("interval time [us]");
```

```
      dataSent = 1;
      elapsedTime = 0;
      senseTest = 0;
      attachInterrupt(digitalPinToInterrupt(sensePin), isr, RISING);//enable sensePin
interrupt. Whenever sensePin receives a rising edge electric signal,
      //the interrupt is triggers and executes the interrupt code. The interrupt
function is called isr.

    }
  }

  if (countMode == 1 && FreqCount.available()) {
    unsigned long freq = FreqCount.read();
    Keyboard.print(elapsedTime);
    Keyboard.set_key1(KEY_TAB);
    Keyboard.send_now();
    Keyboard.set_key1(0);
    Keyboard.send_now();
    Keyboard.println(freq);
    Serial.println(freq);
  }

  if (countMode == 2 && dataSent == 0) {
    //if in interval counting mode and new time interval data has been obtained but
not yet sent (dataSent=0)
    Keyboard.println(bangTime);
    dataSent = 1;//new data is sent, so the logic becomes true
  }

  if (countMode == 2 && elapsedTime > senseLength) {
    //if in interval mode and time elapsed exceeds the time intended for data
collection
    detachInterrupt(digitalPinToInterrupt(sensePin));//disable interrupt of sensePin
    digitalWrite(intervalLED, LOW);//indicate that interval counting mode stopped
    countMode = 0;//shows that we are in nothing mode
  }

}

void isr() {
  //when sensePin receives a rising edge electric pulse, the code is executed.
  bangTime = senseTest;//set time taken for a pulse to be detected as senseTest, which
is the time elapsed
  senseTest = 0;//clear the time elapsed between detection events
  dataSent = 0;//since new data of time interval is acquired, we change the logic of
dataSent to false 0 so that the code will send this data.
}
```

Appendix B: A bit more on Semiconductor Device Physics and Avalanche Breakdown

A SPAD uses a reverse biased p-n junction as the photon detector – in this case an LED. At the appropriate bias voltage, no current will flow 'backward' through the LED when no light is incident on the junction, but the absorption of a photon will produce an electron-hole pair that is quickly separated by the large electric field in the junction. The electrons are accelerated by the field and collide with other atoms, which produce additional excited electrons. The process continues (see figure), building up an 'avalanche' of electrons that produces a current/voltage pulse that is large enough to be measured. This process is analogous to the behavior of a Geiger-Müller tube.

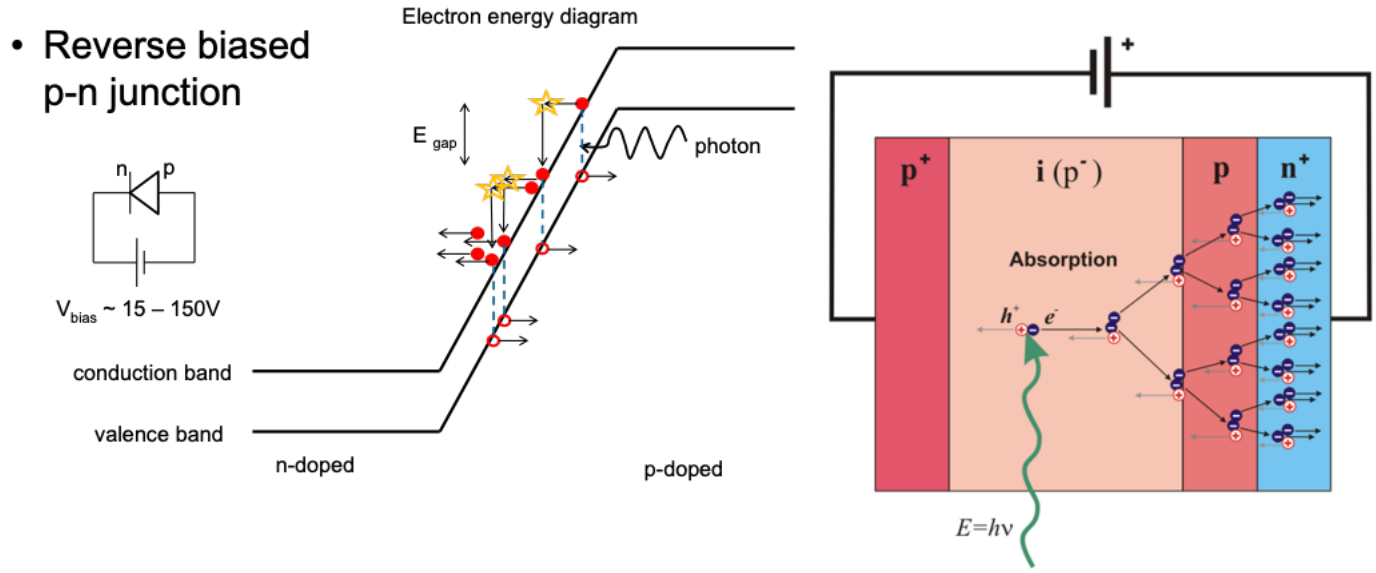


Figure 2. Electron energy diagram in a reverse biased p-n junction. An absorbed photon creates an electron (solid circle) and a hole (empty circle). The large reverse bias voltage produces an electric field in the junction that accelerates the electrons toward the n-doped side. When an electron collides with an atom, its kinetic energy can be used to excite an additional electron into the conduction band. Inset: diagram of a reverse biased p-n junction. Images licensed under Creative Commons Share Alike 3.0.

"Two mechanisms can cause breakdown, namely avalanche multiplication and quantum mechanical tunneling of carriers through the bandgap. Neither of the two breakdown mechanisms is destructive. However heating caused by the large breakdown current and high breakdown voltage causes the diode to be destroyed unless sufficient heat sinking is provided."

"When applying a high electric field, carriers gain kinetic energy and generate additional electron-hole pairs through impact ionization."
http://ecee.colorado.edu/~bart/book/book/chapter4/ch4_5.htm

"Carriers can be generated in semiconductors by illuminating the semiconductor with light. The energy of the incoming photons is used to bring an electron from a lower energy level to a higher energy level. In the case where an electron is removed from the valence band and added to the conduction band, an electron-hole pair is generated. A necessary condition is that the energy of the photon, E_{ph}, is larger than the bandgap energy, E_g. As the energy of the photon is given off to the electron, the photon no longer exists." http://ecee.colorado.edu/~bart/book/book/chapter2/ch2_8.htm#2_8_6

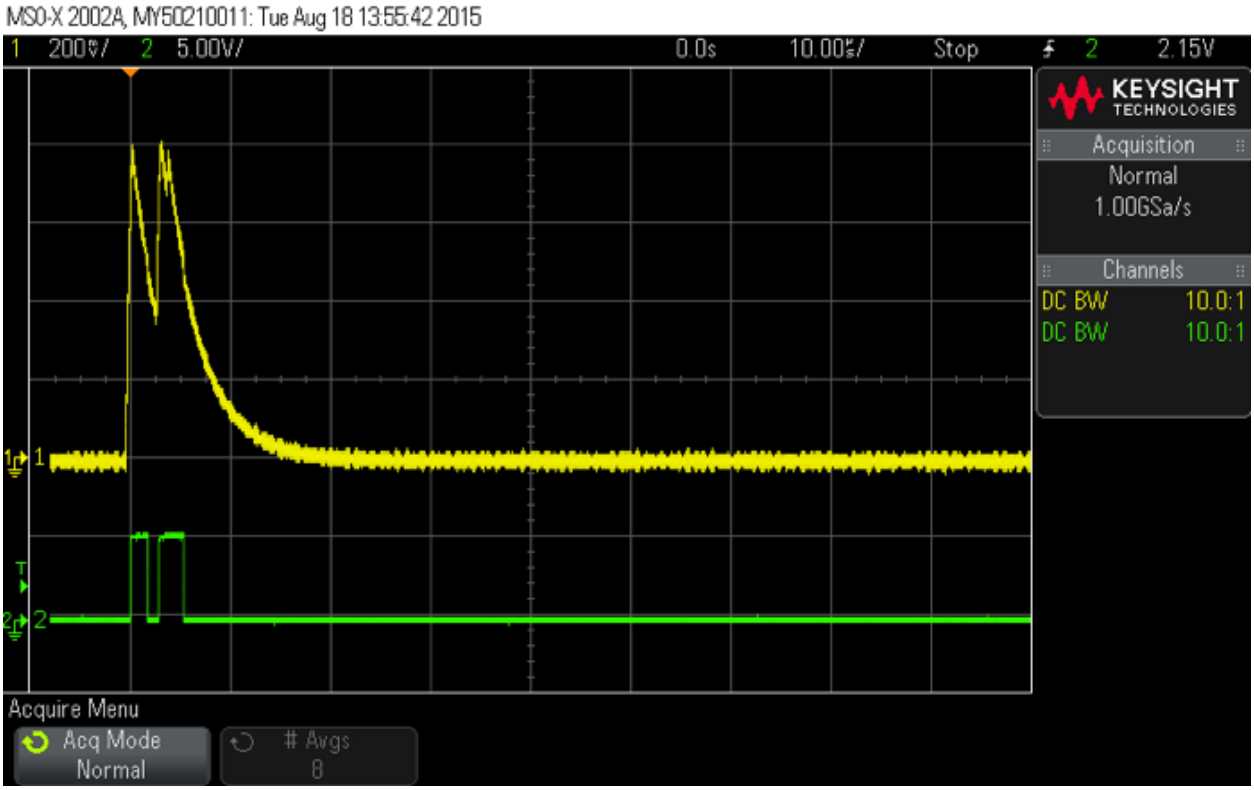
Appendix C: *Artifacts & Errors* Associated with Avalanche Photodetectors

Pulse widths and *dead time*

By varying the value of the resistor you place in series with the LED/SPAD, the *RC* time constant of the circuit can be tuned. This effectively **controls the *dead time* of the detector** (the time during which the detector cannot register the arrival of another photon).

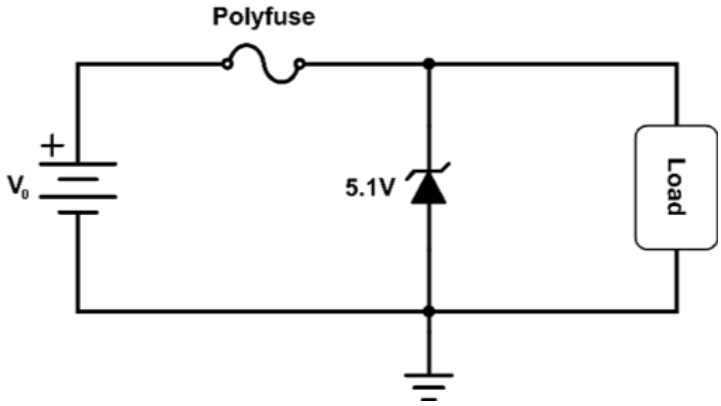
After-pulsing

One crucial pathology seen in SPAD detectors is after-pulsing, which is a second pulse that occurs before the dead time has elapsed, as shown below. Such pulses do NOT correspond to the detection of a second photon (which we do not expect to occur until the unstable quiescent state is reestablished). Instead, after a photon initiates the first avalanche, it is possible for an electron inside the p-n junction to be left in an excited state, and if that electron gains enough energy (usually thermal energy) to reach the conduction band, a second avalanche will occur, *without a photon initiating it*. Thus, a *single* photon can cause not one, but a sequence of two or more pulses that are *not independent* and that occur over very short time intervals. After-pulsing can show up in the data as *non-Poisson* distributions (since some of the pulses are no longer random), and as a large spike at short times when examining the time between pulses. The "high-end" SPADs we use in your later coursework display after-pulsing (usually MUCH) less than 4% of the time, but the "cheap" SPADS you use for your initial investigations offer no such guarantees.



Appendix D: Extra Steps one can take for Circuit Protection

A simple circuit is used to protect the Teensy microcontroller and the +5V USB power supply from excessive current and voltage.

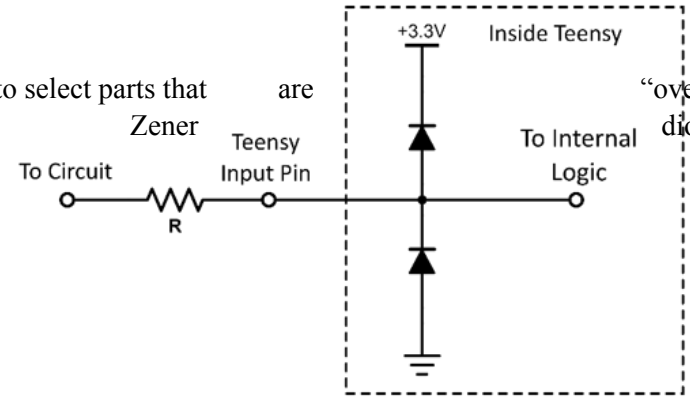


USB ports are designed to supply 2.5W of power at a nominal voltage of +5V. This +5V supply is used to power the microcontroller and comparator circuits, the combination of which draws around 50mA (most of the current is drawn by the Teensy). If a low-impedance load (such as a short circuit!) is presented to the supply, the host computer may be damaged. To prevent this, a 500mA PTC resettable fuse is inserted in series with the +5V supply rail. PTC stands for *Positive Temperature Coefficient* – current flowing through the fuse causes it to heat up, and as it heats the resistance increases, effectively limiting the current. Unfortunately this is a relatively slow process and dependent on the overcurrent value. For the Bourns MF-R025 500mA polyfuse, a 1A overcurrent takes about four seconds to trip, whereas a 2A overcurrent takes about 500ms. Most modern computers and power supplies are short-circuit protected, so the fuse as overcurrent protection is (hopefully) a redundancy.

More critically, we want to protect both the relatively expensive Teensy and the USB power supply from overvoltages. The +5V USB supply feeds a LP38691 3.3V voltage regulator which has a maximum input voltage of +10V. In the event a voltage of greater than 5V is present on the supply rail, a reverse-biased 5.1V Zener diode in parallel with the load limits the voltage to around 5.1V. By itself, the Zener diode would draw as much current as the external power supply could source, and unless it were a particularly beefy Zener, would cease to provide protection due to its thermogenic demise. Upon destruction, the Zener would fail open-circuit, allowing for further destruction of the Teensy and possibly the host computer. To prevent this, the polyfuse serves to limit the amount of current flowing through the Zener in an overvoltage condition. When the Zener diode is in reverse breakdown, the voltage across its terminals is about 5.1V and the polyfuse limits the current flowing through it to a maximum of 500mA. We therefore require a Zener with a power rating of:

$$P = IV = 0.5[A] \times 5.1[V] = 2.55[W]$$

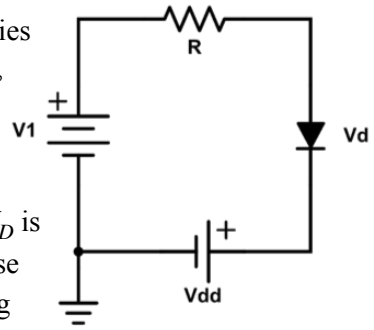
It is good practice to select parts that are "over-specified in power," so here a 5W 1N5338 Zener diode was chosen.



It should also be noted that the Teensy has a maximum voltage rating of 5.5V and a minimum voltage rating of -0.3V for each of its input pins. Connecting a +25V power supply directly to any of the Teensy's pins will *release the magic smoke!* The Teensy does have internal input protection diodes, as shown above, that provide some protection from low-current voltages outside this -0.3V to +5.5V range.

To see how this operates, assume you have connected a voltage of greater than around 3.6V directly to the Teensy Input Pin. Because 3.6V - 3.3V = +0.3V is greater than the voltage needed to place the upper diode into forward conduction, the diode conducts current and the voltage going to the internal logic circuitry is limited to 3.6V. Likewise if a voltage lower than around -0.3V is placed on the Teensy Input Pin, the diode connected to ground goes into forward conduction and the voltage presented to the internal logic is limited to -0.3V.

These internal protection diodes are only rated to 10mA, so to protect from low-impedance overvoltages, such as those from a power supply, one can include a series resistor to limit the current. Kirchoff's Voltage Law for the circuit shown below is,



$$0 = -V_1 + V_R + V_D + V_{DD}$$

Where V_1 is the power supply voltage, V_R is the voltage drop across the resistor, V_D is the voltage drop across the diode, and V_{DD} is the positive power supply (in this case 3.3V). We want to find the resistance necessary to limit the current drawn, so using Ohm's Law and rearranging yields,

$$R = \frac{V_1 - V_{DD} - V_D}{I}$$

Many lab power supplies can source up to 30V, so we choose $V_1 = 30V$. I suspect that the internal diodes are *Schottky diodes*, which have a forward voltage of around $V_D = 0.3V$. $V_{DD} = 3.3V$ and $I = 10mA$ so,

$$R = \frac{30V - 3.3V - 0.3V}{10mA} = 2.64k\Omega$$

To be safe, choose a resistor of 10kΩ as an input resistor.

Circuit protection is a topic unto itself, and far more sophisticated circuits are beautifully and concisely discussed in *The Art of Electronics*, 3rd ed. Section 9.13.

See my notes from when I did the lab with Jonathan Newport

See notes from Mark Masters on the PSoC version

Insert the commented version of the code, from Henry Minzhao Liu