Inexpensive Single Photon Detection Circuit with AND114R LEDs

Spandan Anupam
GitHub

School of Physical Sciences
National Institute of Science Education and Research

February 3, 2021

1. EXPERIMENTAL APPARATUS

1) LED (AND114R)
2) Comparator (LM311)
3) 10kΩ Potentiometer
4) $100-900$kΩ Resistors (3×1kΩ, 2×1MΩ)
5) $1-20$kΩ Resistors
6) Arduino UNO
7) Jumper wires
8) Breadboard(s)
9) Multimeter
10) Constant potential source (24V, 15V, 5V)
11) 3D printer and filament

2. THEORY

For this experiment, we will be following several references, but the most notable of them is (1).

I *Avalanche Photodiodes*

We normally use avalanche photodiodes for single photon experiments, given that detection of a singular photon turns out to be really hard with a normal photodiode. To understand how these work, we may look at their analog, photomultipliers. APDs are considered the semiconductor analog of photomultipliers, used in nuclear counting experiments.

The way these work is, on a high reverse bias voltage, something called *impact ionization* starts to happen, where after one electron has been generated, it starts accelerating and bumping into other electrons, and making them free. This process continues for a while and makes a detectable current, which we can pass through a comparator to get rid of the noise. The process of current multiplication here is called *avalanche breakdown*, which is used for these diodes.

But given the sensitivity of these, we could have guessed that they cost a lot. For student experiments here, we can replicate a similar effect with a AND114R LED, which is comparatively inexpensive, and easy to find. The idea here is not to find a single photon, but is to detect even a single photon when flooded with light, given the inefficiency of these diodes.

There will come a time when due to the Joule heating caused due to the flow of current through the diode, an extra avalanche might be triggered while one is happening. This is known as *afterpulsing*, and will lead to a higher count of photons in our readings.

II *Comparator*

Comparators are pretty easy to understand, for us, they are just an OpAmp circuit with two inputs, with a `True` output if inverting input is greater than the non inverting input and `False` otherwise. For our case, we are using a LM311N comparator IC, whose datasheet can be found in (2).

For our case, we are using the comparator to mellow the noise from the background. Given that we are working in a relatively open environment, we have a lot of noise ready to drown our actual data. We can set a cap on what we are allowing and what we are not, using a comparator and a voltage divider circuit as shown in the diagram. We can then connect this circuit to our reverse biased LED to complete the circuit needed.

III *Circuit*

We have the circuit, as discussed in the sections above here. We can see here, that we can have two classes of resistors that are used.

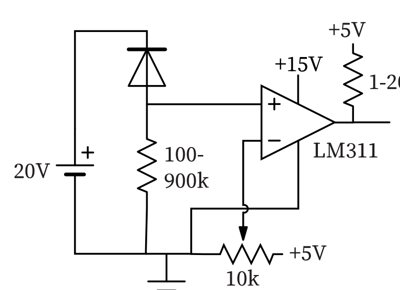


Figure 1: Detector circuit, made with the help of ref 1

The quenching resistor, and the pull up resistor. Changing the quenching resistor will change the pulse width, essentially deciding when the pulse is "quenched". The pull up resistor however, "pulls up" the output voltage. Since we need a 5V output for our detector, here, the internal ADC of the Arduino board.

It is to be noted that instead of the Expeyes and KuttyPy boards, I have been able to modernize the experiment a bit and compress the whole circuit into a single Arduino board, using two small snippets of code that will be mentioned in Section 6.

IV *Counting Statistics*

APDs are quite similar to Gieger counters that we use in nuclear physics experiments. They are even similar in the fact that the triggered events are random and will follow certain statistics, for our case, it is gaussian, considering the number of events that we are counting. So, within the error limits, we should be seeing some curve like:

$$p(x, a, b, c) = a \times e^{(x-b)^2/2c^2} \tag{1}$$

Where we have allowed a, b, c to be the fit parameters. We will see soon, in the Section 4 that this relation holds and we indeed have a gaussian distribution, some prettier than the others.

Au contraire, when we are expecting smaller, random counts (really small intensities), where counts in time are independent from each other (exactly the same condition), we *could* see a poisson distribution. It depends on a single parameter a, and is given by:

$$p(x, \lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{2}$$

We can show that gaussian distribution is just an approximation of the poisson distribution in the limit of a high mean, here, a. Using Eq 2, when we have $x = \lambda(1+\delta)$, at high λ, and a small δ, we will see a transition from discrete $p(x)$ to a continous p. In that limit, we can use the stirling approximation as:

$$x! = \sqrt{2\pi x} e^{-x} x^x \tag{3}$$

Plugging this in, we will get something like:

$$\begin{aligned} p(x) &= \frac{\lambda^{\lambda(1+\delta)} e^{-\lambda}}{\sqrt{2\pi} e^{-\lambda(1+\delta)} [\lambda(1+\delta)]^{(\lambda(1+\delta)+1/2)}} \\ &= \frac{e^{\lambda\delta}(1+\delta)^{-(\lambda(1+\delta)+1/2)}}{\sqrt{2\pi\lambda}} \\ &= \frac{e^{-(x-\lambda)^2/(2\lambda)}}{\sqrt{2\pi\lambda}} \end{aligned} \tag{4}$$

The full proof from the step 2 to 3 has been given in (3). Keeping that aside, this tells us that a higher number of counts, we should be seeing a gaussian distribution instead of poisson, which we are, as shown later.

While we see those measurements, we got another curve to look at. In random distributions, the probability distribution of time between events is given by an exponential, and the time shift is caused due to the dead time. So, we get the equation as:

$$p(t) = r \times e^{-r(t+\tau)} + c \tag{5}$$

Here, τ will give us the dead time of the detector, which we will calculate using our experimental data.

3. MODELLING

Tests in diffused lighting showed me that it was essential to shield the detector from the background lighting. Given the high level of noise these dectors are capable of detecting, I designed a simple walled box to block out most of the light. The idea behind this design is pretty simple, the mode the pin holes, the straighter the light ray has to be, to pass through them. And, putting a mirror to make the arrangement at 90° allows me to cut through even more stray light. The effect is visible in the measurements. Before putting the apparatus in the box, even a dark room would give me a gaussian distribution from the noise itself. The post-box background reading speaks volumes about what the effect of this little 3D printed contraption was.

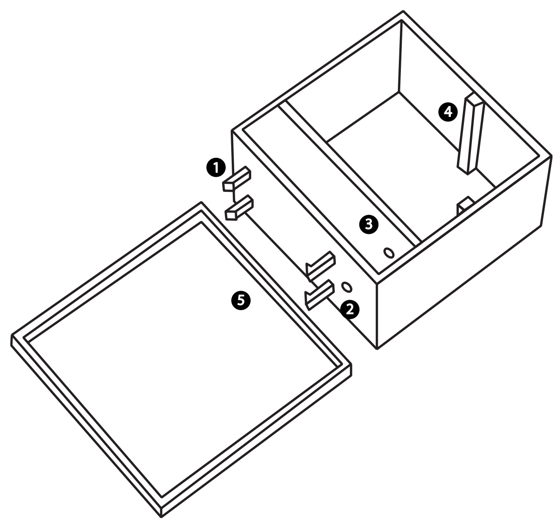
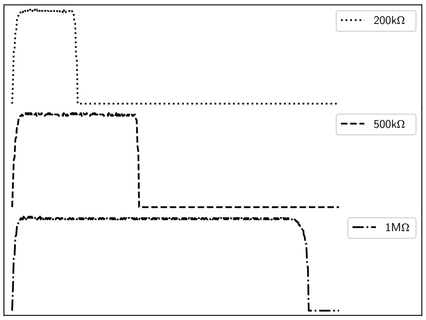


Figure 2: 3D blender model of the diode casing. The labels are given as: (1) Breadboard clips, (2) LED holder and light holes, (3) Wall to prevent light getting near the diode, (4) Mirror bracket and (5) Box cover

A bit about the setup: I made clips and allowed for manual alignment and adjustment. On second thoughts, allowing for a fabry-perot style adjustment system would have been better, but this system does the job. The breadboard clips onto the holding spot and the LED goes into the hole, and is held on using a two part epoxy. The mirror bracket allows is to push fit a mirror at exactly 45°. After printing, I blacked out the whole interior to prevent diffused light from getting in.

4. PLOTS

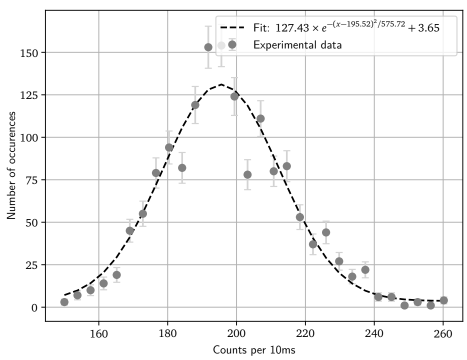
The first plot we will be looking at will be the pulse shapes for different values of the quenching resistors.

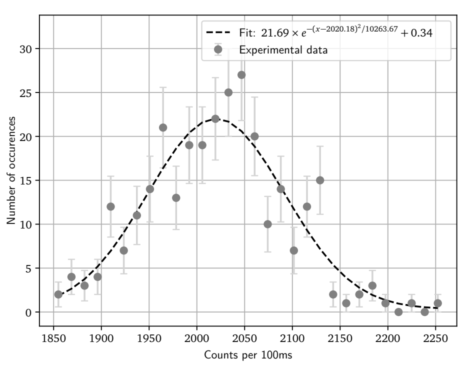


I took the liberty to clean out the base of all the signal pulses, because they are unimportant. This plot clearly shows that as width $\propto R$. Maybe not linearly, but it does change with the resistance.

I *Open lighting*

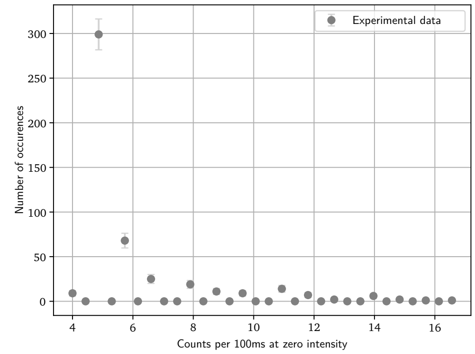
These plots are for readings taken for number of pulses per 10 and 100ms in diffused, open lab lighting.





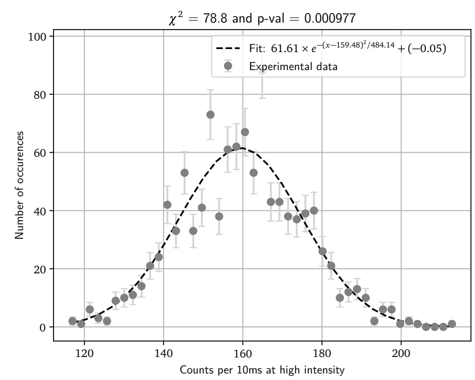
II *Laser background*

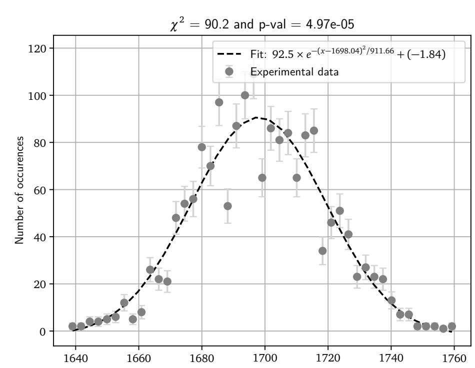
These were the readings taken in the closed enclosure, with a collimated laser beam.

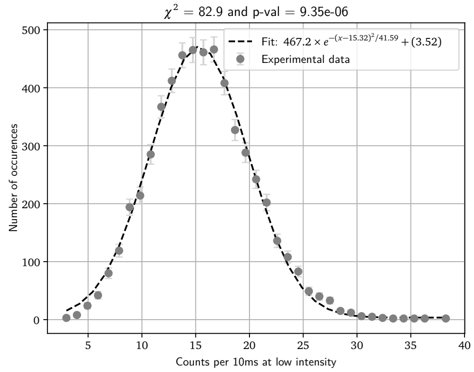


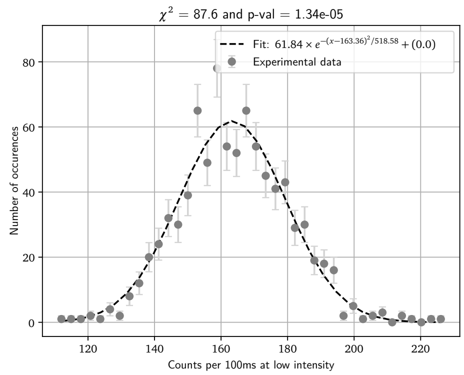
The noise drops straight to zero, except for the spike at 5, which seems to be the background due to some radiation. This reading comes to use later, because we had to clean up our data for some other readings later, when we had a consistent low background count for all of our time slots.

III *Laser at two intensities*

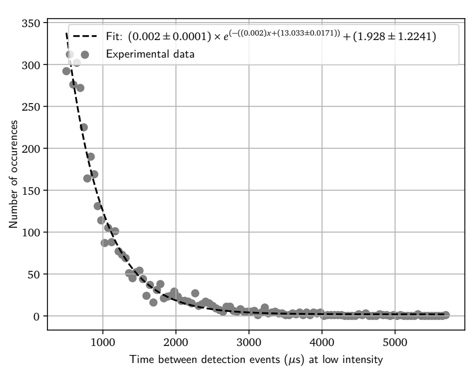








IV *Time between pulses*



We see that the graph fits fairly well with an exponential model, and the dead time can be calculated as $6516.5 \pm 325.82\mu s$. The gaussian fitting results show us a χ^2 value of around 80 and an extremely low p-value (in the order of 10^{-5}), which to my understanding, does indicate that our assumption of the distribution being gaussian is correct.

5. CONCLUSION

To conclude, I would say that the experiment was a success, with multiple readings showing a gaussian distribution. We were able to make a setup that is able to generate these results with fairly good repeatability and a commendably low noise to signal ratio. Though I had to clean up *some* of the data, that was expected with this kind of a setup, and the process has been referenced in multiple sources. Also, using the usual exponential decay curve in `scipy.optimize` and `python`, we were able to calculate the dead time to be $6516.5 \pm 325.82\mu s$, and from multiple readings, saw that it stayed in the 3–7ms range, which is less than our minumum time slot of 10ms.

It is to be noted that the curve being fit with a gaussian is all that matters, and the FWHM in this case, would just represent the intensity of the radiation incident on the detector. The height of the "histogram" points will depend on the bin size, and will have no real effect on the curve that we get. We can see that be it a higher, or lower intensity, it doesnt affect the curve being a gaussian, and that's all that matters.

6. APPENDIX

Just two small snippets of C++ code were used to run the arduino according to our requirements. All of the code has been uploaded ready for use in my GitHub.

The first code here is for counting the number of pulses in a set time. In this example, the interval has been set to 100ms.

Listing 1: num_pulses.ino

```
int pin = 7;
unsigned long period;
unsigned long timeRes = 100;
int numPulses = 0;
unsigned long timeStart = millis();

void setup() {
        Serial.begin(9600);
        pinMode(pin, INPUT);
}

void loop() {
        if (millis() - timeStart >= timeRes) {
                Serial.println(numPulses);
                numPulses = 0;
                timeStart = timeStart + timeRes;
        }
        period = pulseIn(pin, HIGH);
        if (period > 0) {
                numPulses++;
        }
}
```

The second snippet here measures the time difference between the dropping and rising edge of a pulse, in μs.

Listing 2: time_difference.ino

```
int pin = 7;
unsigned long duration;

void setup() {
        Serial.begin(9600);
        pinMode(pin, INPUT);
}

void loop() {
        duration = pulseIn(pin, LOW);
        Serial.println(duration);
}
```

Both of these can be compiled and run on the Arduino, and possibly on other ATMEGA chips after some tweaking.

Apart from that, to give an idea of the setup involved to get the readings that we need, I have attached an image of the full setup that we are using, on the optical breadboard table.

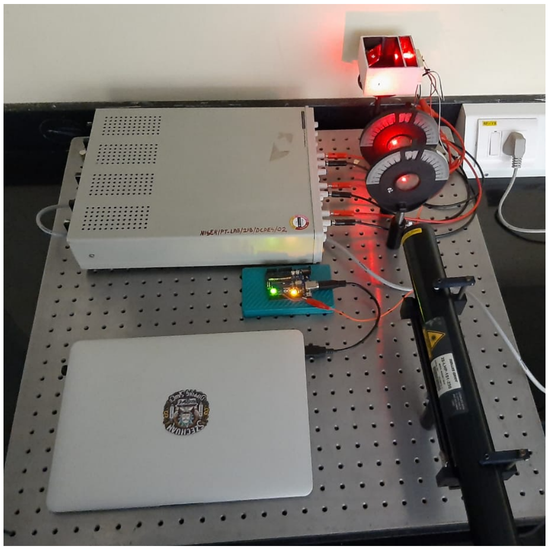


Figure 3: Full view shot of the setup with the top open

7. ACKNOWLEDGEMENTS

I would like to extend a special thanks to Prof. Subhankar Mishra for lending me his 3D printer for use in this project. I would also like to thank Prof. Kartik Senapati and Dr. G Santosh Babu for their help during the experiment.

8. REFERENCES

1) L. I. McCann, BFY Proceedings, 64–67 (2015)

2) Phillips, LM311N component datasheet, (https://www.alldatasheet.com/datasheetpdf/pdf/17875/PHILIPS/LM311N.html)

3) Gaussian distribution as a limit of the Poisson distribution, 2012, (https://www.roe.ac.uk/japwww/teaching/astrostats/astrostats2012_part2.pdf)