CLAYTON STATE UNIVERSITY

Design and testing of photon-based hardware random number generator

By: Nikolas Thornton*, Dmitriy Beznosko

*presenter.

Random numbers & their uses

CYBERSECURITY

- Key generation
- Encryption algorithms

MACHINE LEARNING

- Model initialization
- Mutations
- Dropout selection

As well as physics simulations, lotteries, and video games.

Pseudo-random number generators

Seed – an input that PRNGs use to define their randomness.

Figure 1. The seed-input box for the video game Minecraft.

Pros:

- Easily replicable results
- No external hardware necessary

Cons:

- Easily replicable results
- Not random enough for many use cases

Photon-based solution to random number generation

Advantages of photon-based solutions

- Photon amplitude detection is an inherently chaotic process [1].
- The hardware involved is relatively cheap.

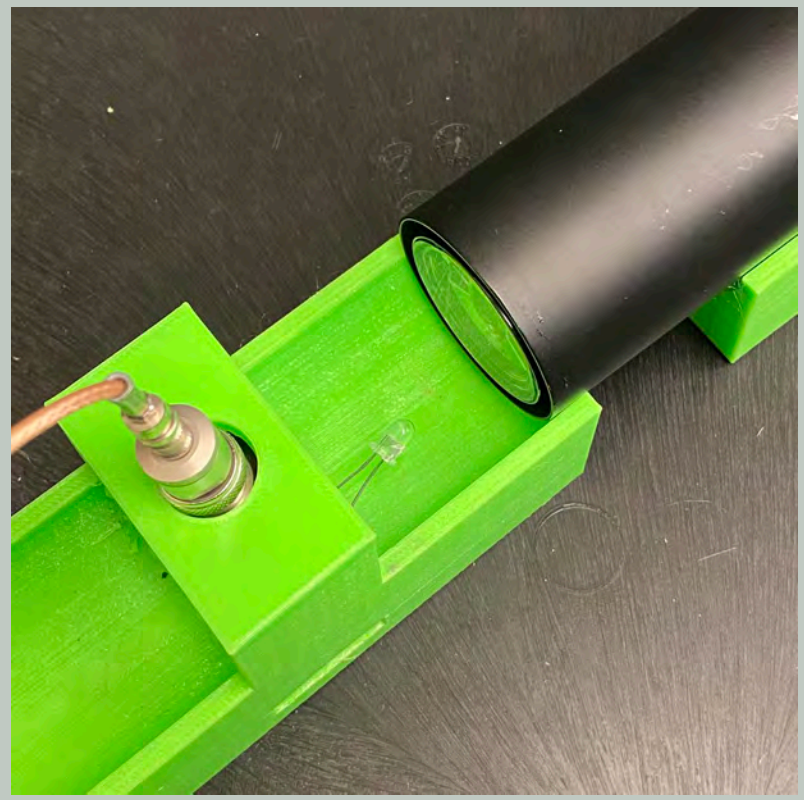


Figure 2. Concept design using an LED that pulses in sync with the reading of a photo-multiplier-tube.

Photon-based HRNG Prototype

Current prototype produces ~3500 amplitudes per second. Amplitudes are sent via serial and saved to a computer where they are processed.

Prototype hardware:

- Arduino Nano
- Multi-Pixel Photon Counter (MPPC) [2]
- Serially programmed power supply
- LED

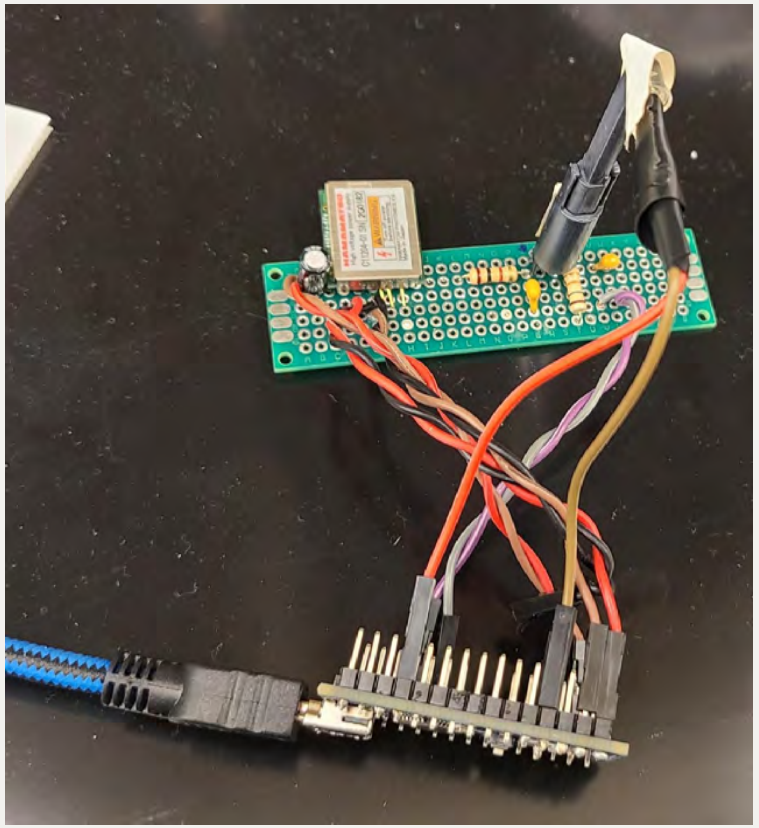


Figure 3. Concept design using an LED that pulses in sync with the reading of a photo-multiplier-tube.

Processing Amplitudes

High/Low

```
if amp[i] > amp[i-1]: bitstream += 1
if amp[i] < amp[i-1]: bitstream += 0
else: skip
```

Even/Odd

```
if amp[i] % 2 == 0: bitstream += 1
else: bitstream += 0
```

Von Neumann's Procedure

- Search bitstream for pairs of 00 and 11.
- For each pair of 00 and 11, add a 0 to a new bitstream A and a corresponding 1 or 0 to bitstream a new B.
- For each non-matching pair, add a 0 to bitstream A and a 1 or 0 (depending on the first bit of the pair) to a new bitstream C.
- Recursively return the value of Neumann's procedure for each bitstream A, B, and C in the order of $C + A + B$.

Average & Standard Deviation Test

Using data collected over the course of 1 minute.

Pre-Neumann bitcount: **214676**
Post-Neumann bitcount: **208707**

This test is sensitive to data that has an inordinately unequal number of 1s and 0s, although it does not determine whether the bitstream is necessarily random.

Pre-Neumann

Average: 0.4987422907078574
Standard Deviation: 0.4999984181648342 6

Post-Neumann

Average: 0.5004000824121855
Standard Deviation: 0.4999998399340378 5

Monte Carlo Pi Estimation Test Results

Using data collected over the course of 1 minute.

Pre-Neumann bitcount: 214676 yielding 13417 coordinates
Post-Neumann bitcount: 208707 yielding 13044 coordinates

Python also generates pi using its own random library but using the same number of coordinates as Pre-Neumann & Post-Neumann. This helps determine whether a poor-quality pi is due to poor quality data or too few coordinates.

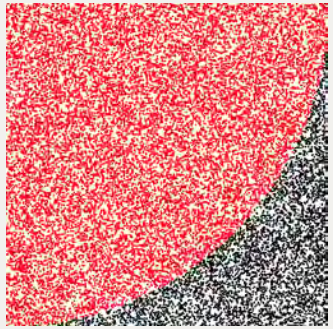


Figure 4. Example visualization of the MCPE test

```
all_dots += 1
if x^2 + y^2 <= 1: red_dots += 1
pi = 4*(red_dots/all_dots)
```

Figure 5. Psuedo-code describing how the MCPE test calculates pi

Pre-Neumann

Estimated Pi: 3.13572333607

Post-Neumann

Estimated Pi: 3.13002146581

Python's coords

Pre-neumman's bit total: 3.13453081911
Post-Neumann's bit total: 3.14872738424

The Fractional Line Symmetry Test

Count how frequently 1s or 0s appear back-to-back for a determined length within the bitstream, called lines, where additional bits at the end of lines are added as fractional lines to the line total both horizontally and vertically for a stacked bitstream.

Terms:

Line: A single instance of back-to-back bits that meet the detect length.

Detect Length: How many times a bit must appear back-to-back before being counted to the line total.

Horizontal Search: Linearly search unmodified bitstream for lines.

Vertical Search: Stack bitstream into a square matrix and count number of lines top to down starting in the top right.

Fractional Line: Additional bits attached to an already full line that get added fractionally to the line total.

Example: Searching the bitstream 100010 **horizontally** for 0s with a **detect length** of 2 totalling 1.5 **lines**.

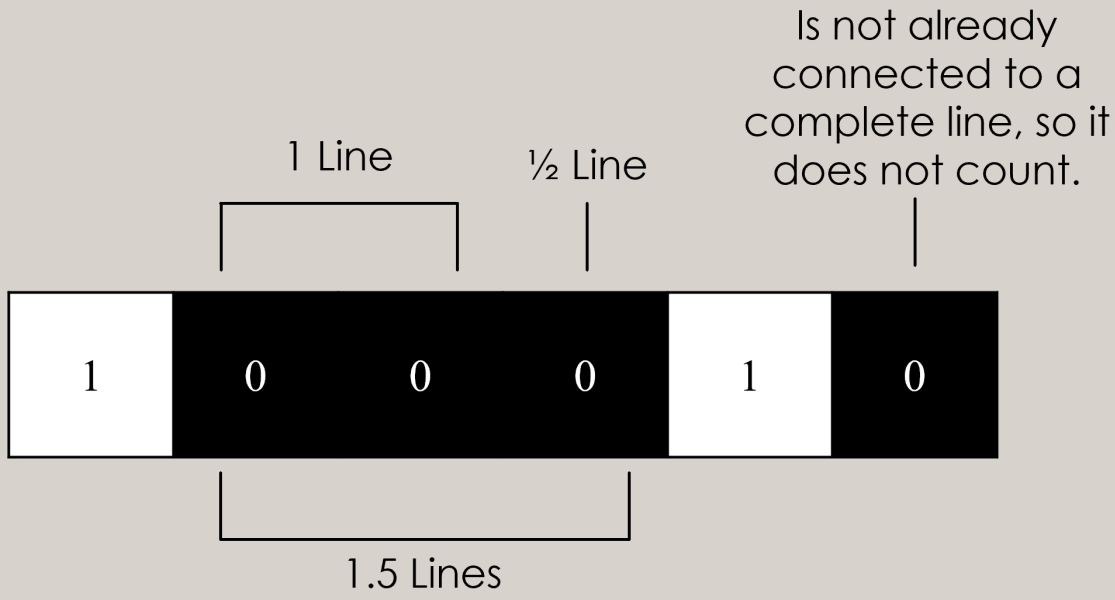


Figure 6. How the FLS test counts lines

FLS Test Visualization

Example Pre-Neumann bitstream visualization using High/Low processing.

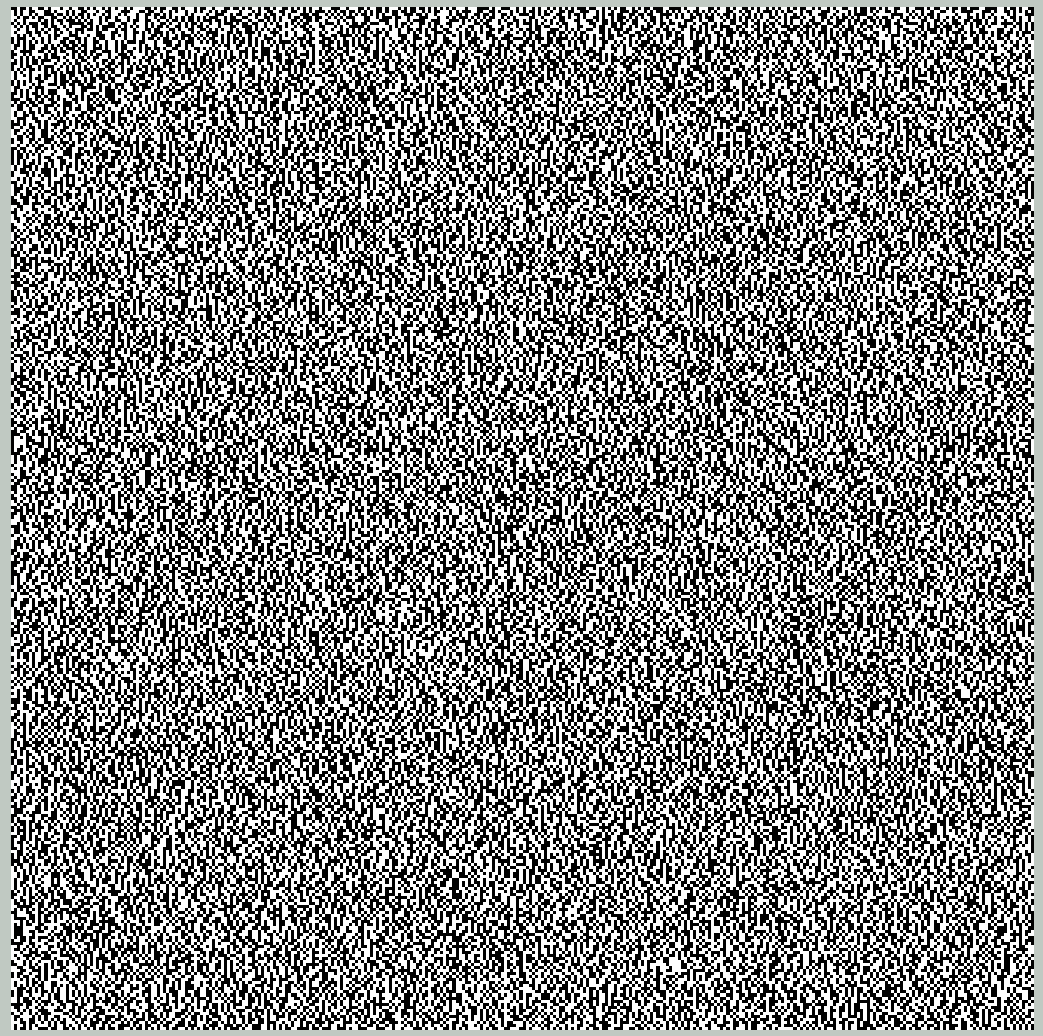


Figure 7. Pre-Neumann High/Low bitstream visualization

FLS Test Visualization

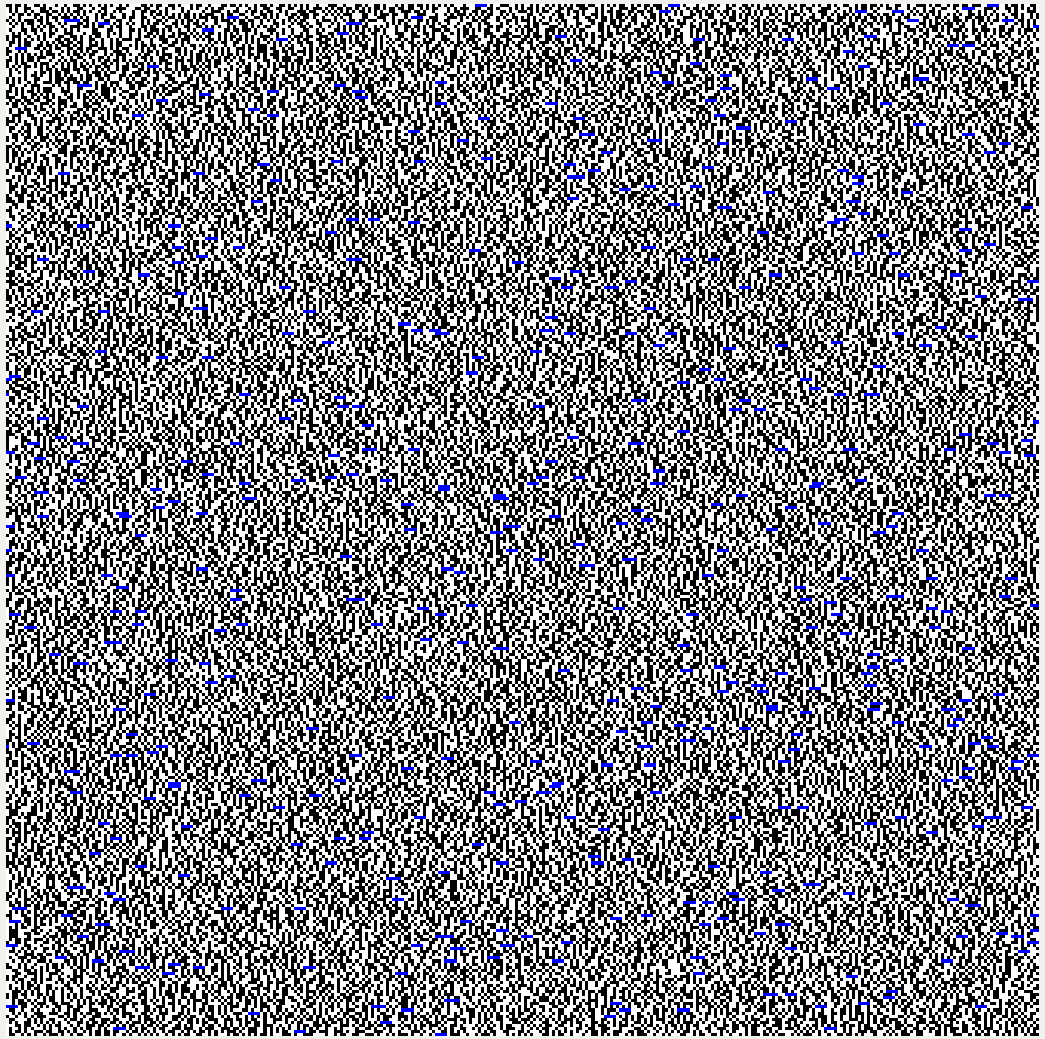


Figure 8. Pre-Neumann High/Low bitstream horizontal FLS test.

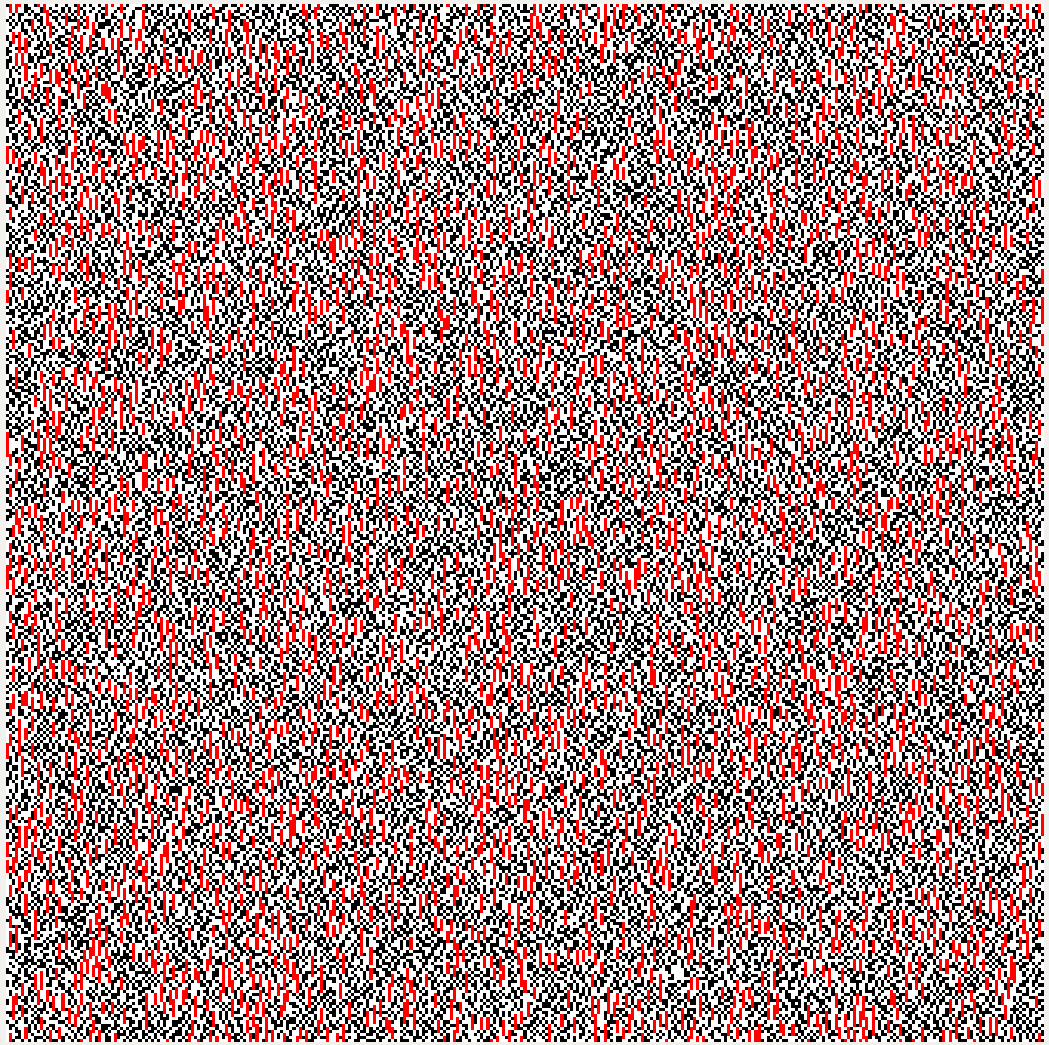
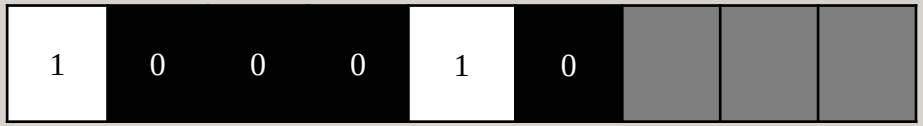


Figure 9. Post-Neumann bitstream visualization vertical FLS test

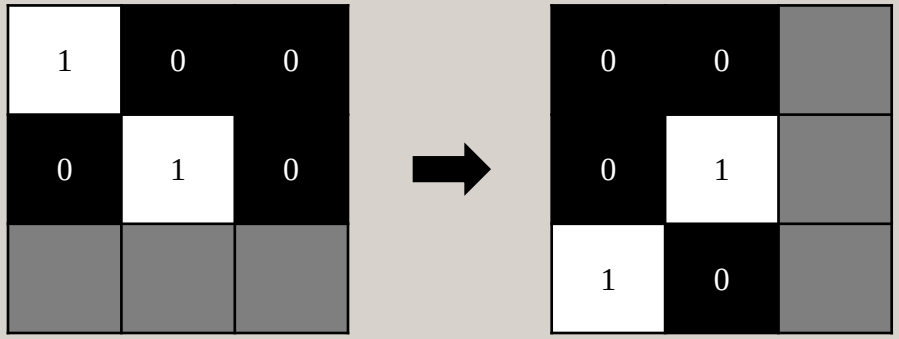
Vertical Line Counting

Original bitstream.

Add "nothing bits" to make bitstream stackable



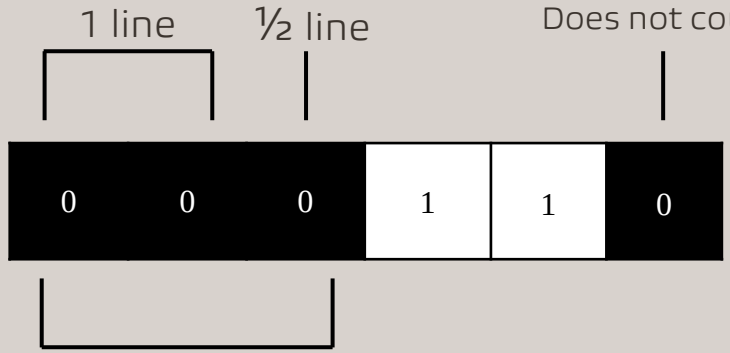
Stack bitstream left to right, top to down into a square image and rotate Counter-clockwise



Unstack bitstream left to right, top to down.

Remove the "nothing bits"

Count lines as you would for horizontal lines



The FLS Test Results

Using data that was collected over the course of 1 minute.

Post-Neumann bitcount: **208707**
Searching for: **0**
Detect length: **4**

Estimated Line Count:	**8152.62**
Horizontal Line Count:	**8094.25**
Vertical Line Count:	**8190.5**

The number of lines expected to be found can be calculated using

$$\frac{n+1}{(2^{n+1})n} \times l$$

where $\boldsymbol{n}$ is the detect length and $\boldsymbol{l}$ is the length of the bitstream.

Visualization

Figure 10. FLS test horizontal line visualization

Figure 11. FLS test vertical line visualization

Conclusion

The photon-based solutions explored in this research have shown great potential as high speed solutions for high quality random number generation.

Future plans:

- Use a faster micro controller solution.
- Implement more tests for randomness such as the Ent test.
- Unify amplitude reading, processing, and testing into one program.
- House the HRNG.
- Secure the LED to the MPPC.
- Use voltage controller for the LED instead of PWM.

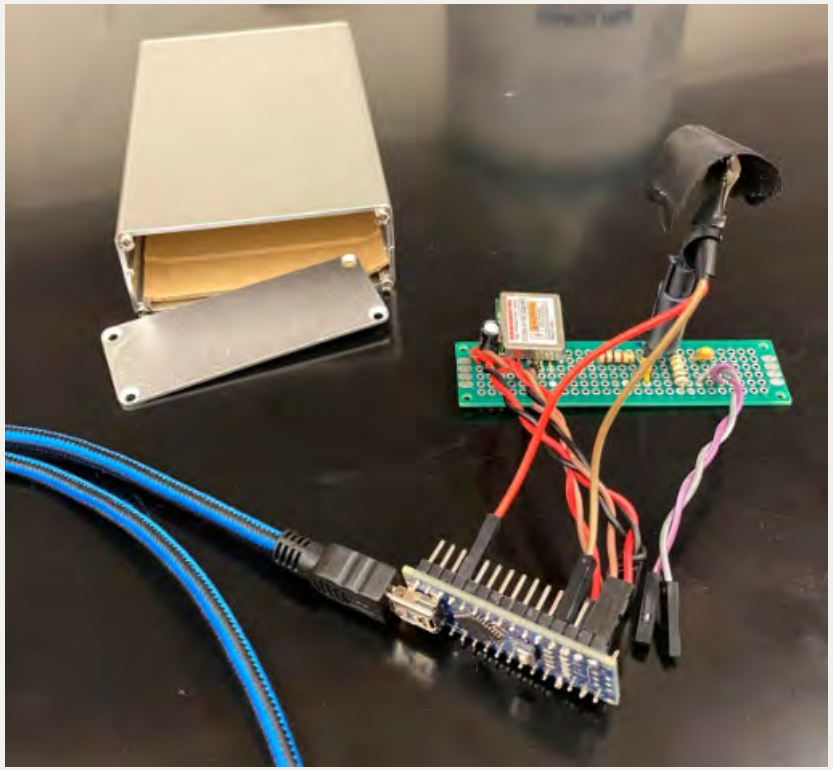


Figure 12. The current HRNG prototype alongside its future enclosure.

References

[1] D. Beznosko et al., "Random Number Hardware Generator Using Geiger-Mode Avalanche Photon Detector", e -Print: 1501.05521, DOI: 10.22323/1.252.0049, PoS PhotoDet2015 (2016), 049

[2] Hamamatsu Photonics K.K., 325-6, Sunayama-cho, Naka-ku, Hamamatsu City, Shizuoka Pref., 430-8587, Japan