

# **Quantum Random Number Generator(QRNG)**

Name: John Regis

Roll Number: 20246718

Submitted to: **Dr Shouvik Dutta**

Professor, Department of Physics, IISER Pune

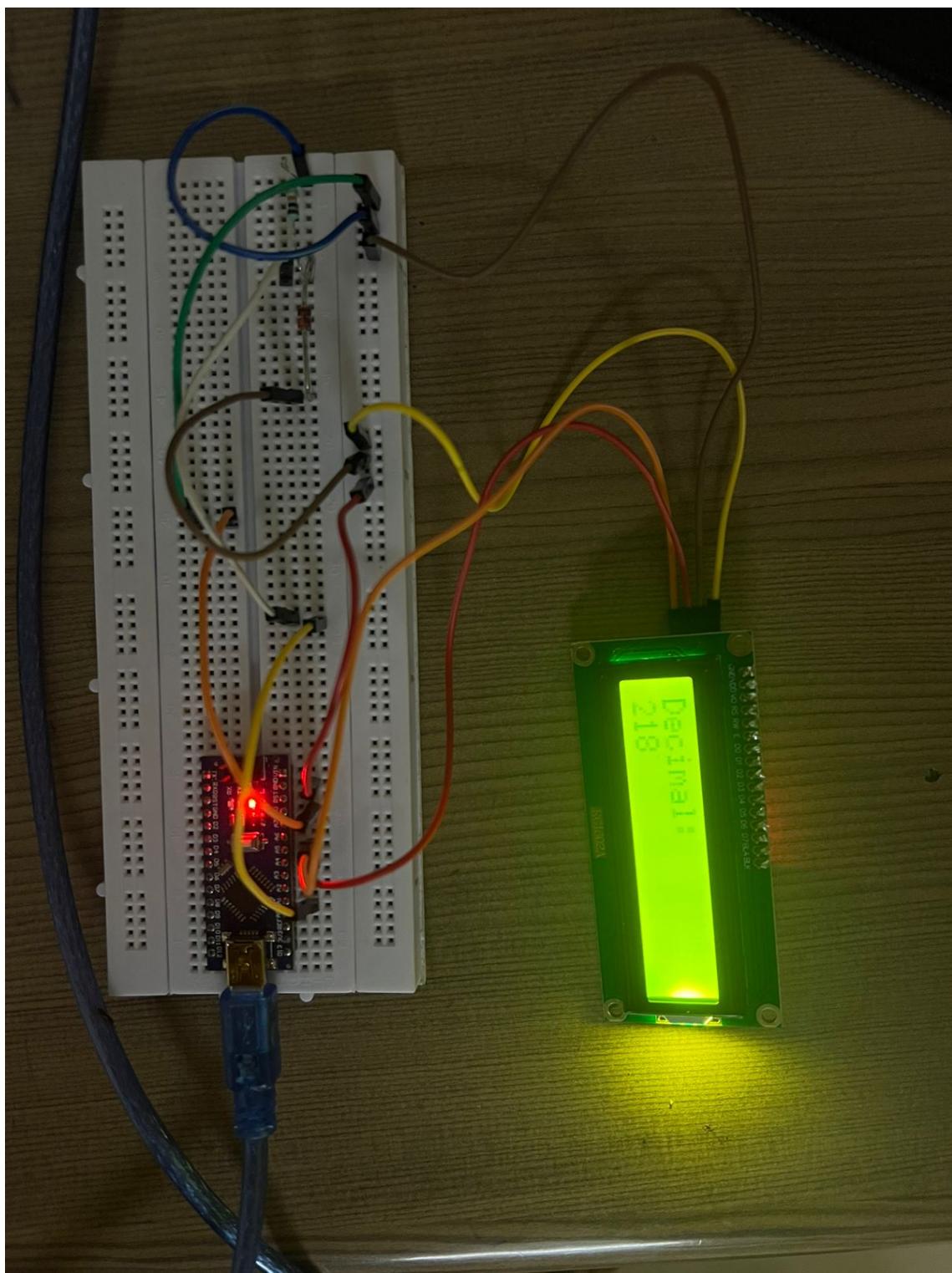
**Abstract.** This project explores a quantum random number generator (QRNG) based on the quantum tunneling phenomenon in Zener diodes. Quantum random number generation provides genuine randomness, critical for cryptographic applications, by harnessing quantum mechanical effects rather than relying on deterministic algorithms. By exploiting quantum tunneling—the probabilistic behavior where electrons pass through the diode's potential barrier—this setup generates unpredictable voltage fluctuations. These fluctuations are amplified, digitized, and processed to create a sequence of unbiased random numbers. Using a configuration of four Zener diodes, this study demonstrates how quantum tunneling serves as an effective entropy source. The generated random numbers show high unpredictability and statistical uniformity, highlighting the viability of Zener diode-based QRNGs for secure cryptographic and computational uses.

.

## 1. Introduction

Classical random number generators (RNGs) are typically algorithmic and pseudo-random, meaning their "randomness" is ultimately predictable given the initial seed. This predictability can compromise the security of systems, particularly in cryptography, where true randomness is essential for encryption strength. To overcome these limitations, quantum random number generators (QRNGs) harness quantum mechanical phenomena, which provide inherent unpredictability due to the probabilistic nature of quantum events.

The motivation for this project stems from the need for a reliable, hardware-based source of true randomness. By exploiting quantum tunneling in Zener diodes, we aim to generate random numbers that are genuinely non-deterministic, contributing to more secure cryptographic systems.



This project report is structured as follows:

1. **Equipment Used** – A detailed list of all components and devices essential for building the QRNG.
2. **Circuit Diagram** – A schematic representation of the setup, showing the arrangement of Zener diodes and other elements.
3. **Theory: Quantum Tunneling** – An overview of the quantum tunneling phenomenon and its application in generating random numbers.
4. **Project Code and Explanation** – The code used to process and interpret the diode-generated data, with an in-depth explanation of each section.
5. **Result and Analysis** - The result and analysis were conducted by performing 100 trials and evaluating the autocorrelation function of the generated random numbers to assess their statistical properties.
6. **Range and Scope of the Project** – A discussion of the limits of this project, including accuracy, randomness quality, and potential applications.
7. **Project Extension** – Suggested improvements and future extensions for enhancing the QRNG, including increased randomness efficiency and integration with larger cryptographic systems.

This report aims to provide a comprehensive understanding of a Zener diode-based QRNG and its practical relevance in secure random number generation.

## 2. Equipments Used

### 2.1. Arduino Nano

Arduino® Nano is an intelligent development board designed for building faster prototypes with the smallest dimension. Arduino Nano being the oldest member of the Nano family, provides enough interfaces for your breadboard-friendly applications. At the heart of the board is ATmega328 microcontroller clocked at a frequency of 16 MHz featuring more or less the same functionalities as the Arduino® Duemilanove. The board offers 20 digital input/output pins, 8 analog pins, and a mini-USB port. The primary processor in the Nano board is the high-performance and low-power 8-bit ATmega328 microcontroller that runs at a clock frequency of 16 MHz. The ability to interface external devices through serial communication supported by the chip with UART TTL (5V), I2C (TWI) and SPI. Nano can be programmed with Arduino software reducing the entry barriers for new users. Smallest dimension embedded hardware makes it a perfect choice for breadboard-friendly projects from the maker community.

### 2.2. Zener Diode

The zener diode is a semiconductor device unique in its mode of operation and completely unreplaceable by any other electronic device. Because of its unusual properties it fills a long-standing need in electronic circuitry. It provides, among other useful functions, a constant voltage reference or voltage control element available over a wide spectrum of voltage and power levels. The zener diode is unique among the semiconductor family of devices because its electrical properties are derived from a rectifying junction which operates in the reverse breakdown region. In the sections that follow, the reverse biased rectifying junction, some of the terms associated with it, and properties derived from it will be discussed fully. The zener diode is fabricated from the element silicon (Si).

Special techniques are applied in the fabrication of zener diodes to create the required properties. This manual was prepared to acquaint the engineer, the equipment designer and manufacturer, and the experimenter with the fundamental principles, design characteristics, applications and advantages of this important semiconductor device.

We used the **1N4733A Zener diode** model in this project because its **5.1V Zener voltage** provides a stable reference, which is ideal for generating the controlled noise needed in our quantum random number generator. This model is also commonly used in labs due to its **1-watt power rating**, which ensures it can handle fluctuations without overheating, making it a reliable choice for educational and experimental purposes.

### 2.3. LCD Display

A **16x2 LCD display** is a small screen commonly used in electronics projects to show text. The "16x2" part means it has **16 columns and 2 rows**, so it can display up to **32 characters at a time** (16 characters on each row). Each character is made up of a grid of tiny dots, allowing letters, numbers, and symbols to be displayed clearly.

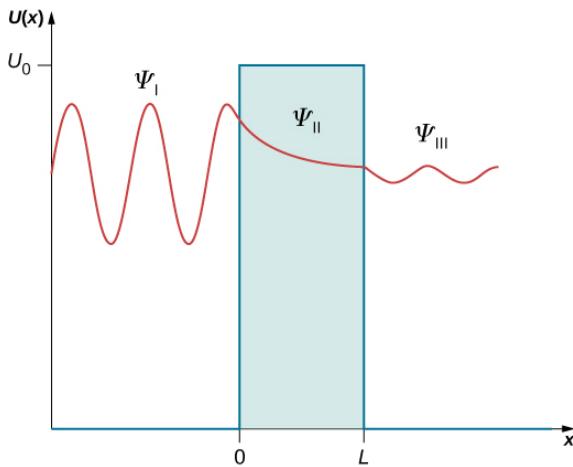
### 2.4. I2C Module

An **I2C (Inter-Integrated Circuit) module** is used to simplify the connection between a micro controller (like an Arduino) and devices such as an LCD display. Normally, a 16x2 LCD requires **at least 6 pins** to connect to a micro controller, which can take up a lot of space and wiring.

The I2C module reduces this by allowing the LCD to connect using only **2 data pins** (SDA for data and SCL for clock) instead of multiple pins. This module uses the I2C communication protocol, which allows multiple devices to share the same two wires, making it easier to add more components to your project without running out of pins.

## 3. Quantum Tunneling

**quantum tunneling**, consider a ball rolling along a surface with a kinetic energy of 100 J. As the ball rolls, it encounters a hill. The potential energy of the ball placed atop the hill is 10 J. Therefore, the ball (with 100 J of kinetic energy) easily rolls over the hill and continues on. In classical mechanics, the probability that the ball passes over the hill is exactly 1—it makes it over every time. If, however, the height of the hill is increased—a ball placed atop the hill has a potential energy of 200 J—the ball proceeds only part of the way up the hill, stops, and returns in the direction it came. The total energy of the ball is converted entirely into potential energy before it can reach the top of the hill. We do not expect, even after repeated attempts, for the 100-J ball to ever be found beyond the hill. Therefore, the probability that the ball passes over the hill is exactly 0, and probability it is turned back or “reflected” by the hill is exactly 1. The ball **never** makes it over the hill. The existence of the ball beyond the hill is an impossibility or “energetically forbidden.” However, according to quantum mechanics, the ball has a wave function and this function is defined over all space. The wave function may be highly localized, but there is always a chance that as the ball encounters the hill, the ball will suddenly be found beyond it. Indeed, this probability is appreciable if the “wave packet” of the ball is wider than the barrier.



A Zener diode utilizes the principle of **quantum tunneling** to conduct in reverse bias when the voltage across it reaches a specific level, known as the **Zener breakdown voltage**. This property is primarily useful in applications where stable reference voltages are required, such as in voltage regulation circuits. Here's how quantum tunneling is involved:

### 3.1. Reverse Bias and High Electric Field:

- When a Zener diode is reverse-biased (with the positive terminal connected to the n-type material and the negative terminal to the p-type), it does not conduct until the reverse voltage reaches a critical value, known as the **Zener voltage**.
- At this voltage, the electric field across the diode becomes extremely strong due to the thin depletion region in a Zener diode's structure.

### 3.2. Quantum Tunneling in the Depletion Region:

- In a strong electric field, electrons in the valence band of the p-type material have enough energy to "tunnel" through the narrow depletion region to the conduction band on the n-type side, despite the energy barrier that would typically prevent this movement.
- This process occurs because of **quantum tunneling**, where particles like electrons have a finite probability of crossing an energy barrier, even if they don't have enough energy to overcome it classically.

### 3.3. Zener Breakdown and Current Flow:

- Tunneling occurs, electrons effectively "jump" across the barrier, allowing a substantial current to flow through the diode even in reverse bias. This current is stable and does not damage the diode, as the Zener diode is designed to handle this kind of breakdown.

## 4. Project code and explanation

### 4.1. Arduino

#### Arduino Code Example

```
1 // Arduino code for generating a binary string and displaying
2 // it on an LCD
3 #include <Wire.h>
4 #include <LiquidCrystal_I2C.h>
5
6 const int analogPin = A0;          // Analog pin for random noise
7 const int ledPin = 13;            // LED pin
8
9 LiquidCrystal_I2C lcd(0x27, 16, 2); // Adjust address if
10 needed
11
12 void setup() {
13     Serial.begin(9600);           // Initialize serial
14     communication
15     pinMode(ledPin, OUTPUT);      // Set the LED pin as an
16     output
17     lcd.begin(16, 2);            // Initialize the LCD for 16
18     x2 characters
19     lcd.backlight();             // Turn on the backlight
20     lcd.setCursor(0, 0);          // Set cursor to the
21     beginning
22     lcd.print("Binary:");        // Print "Binary:" on the LCD
23 }
24
25 void loop() {
26     String binaryString = "";      // To store the binary
27     representation
28     int decimalValue = 0;          // To store the decimal value
29
30     for (int i = 0; i < 8; i++) {
31         int noise = analogRead(analogPin); // Read random
32         noise
33
34         int randomBit = noise & 1; // Extract the least
35         significant bit
36
37         binaryString += String(randomBit); // Append the bit
38         to the string
39
40         decimalValue = (decimalValue << 1) | randomBit; // Update decimal value
41
42         Serial.print("Bit ");
43         Serial.print(i);
44         Serial.print(":");
45         Serial.println(randomBit);
46 }
```

```

37     digitalWrite(ledPin, randomBit ? HIGH : LOW); // Blink
38     LED based on bit
39
40     delay(100);
41
42     // Display binary string on the LCD
43     lcd.setCursor(0, 1);
44     lcd.print(binaryString);
45     lcd.print("_____");
46
47     // Print binary and decimal values to Serial Monitor
48     Serial.print("Binary Number: ");
49     Serial.println(binaryString);
50     Serial.print("Decimal Value: ");
51     Serial.println(decimalValue);
52
53     delay(1000); // Pause before clearing
54
55     // Display decimal value on LCD
56     lcd.clear();
57     lcd.setCursor(0, 0);
58     lcd.print("Decimal: ");
59     lcd.setCursor(0, 1);
60     lcd.print(decimalValue);
61     lcd.print("_____");
62
63     delay(2000);
64 }
```

## 4.2. Code Breakdown

The provided Arduino code utilizes Zener diodes to generate random bits based on quantum tunneling. The program begins by including the necessary libraries: Wire.h for I2C communication and LiquidCrystal I2C.h to control the LCD display using I2C. The analogPin is configured to read the noise generated by the Zener diodes, while the ledPin is used to visually display the generated random bits.

In the setup() function, the serial communication is initialized with Serial.begin(9600) for debugging purposes. The LED pin is configured as an output, and the LCD is initialized with the lcd.begin(16, 2) command, setting it to a 16x2 display. The backlight of the LCD is turned on with lcd.backlight(), and a label "Binary:" is displayed on the screen to indicate where the binary sequence will appear.

The loop() function runs continuously, first initializing two variables: binaryString to store the binary sequence and decimalValue to hold the decimal equivalent of the binary sequence. A for loop then reads the noise from the Zener diodes via analogRead(analogPin). This noise value is processed to extract the least significant bit (LSB), which is then treated as a random bit (randomBit). The random bit is appended to the

`binaryString`, and the decimal value is updated accordingly by shifting the current value left and adding the new bit.

For each random bit generated, the bit is printed to the Serial Monitor along with the bit number, and the LED is toggled to represent the bit visually, where HIGH corresponds to a bit of 1 and LOW to a bit of 0. A short delay of 100 milliseconds is added to make the process visible. Once all 8 bits have been generated, the complete binary string is displayed on the second line of the LCD, and the string is also printed to the Serial Monitor, along with its decimal equivalent.

After a 1-second delay to allow the user to view the output, the LCD screen is cleared, and the decimal value is displayed. Another 2-second delay is added before the process repeats. This cycle continues, generating a new random number every 2 seconds.

This system leverages the inherent unpredictability of quantum tunneling in Zener diodes to generate high-quality random numbers, suitable for cryptographic or other applications requiring randomness

## 5. Result and Analysis

### 5.1. 100 Trials

100 trials were conducted to perform a result analysis and verify whether the generated numbers are truly random by evaluating their statistical properties through the autocorrelation function. The output is as follows

---

236 216 234 98 192 156

---

160 91 23 109 117 138

---

255 195 171 77 94 154

---

194 214 219 201 27 212

---

134 118 33 226 8 194

---

255 201 59 213 246 54

---

188 218 128 91 90 142

---

134 18 82 74 24 63

---

5 7 19 227 182 37

---

221 233 73 93 97 87

---

205 129 25 235 171 240

---

171 224 49 111 75 221

---

239 9 128 113 246 184

---

190 166 237 1 33 202

---

12 7 54 119 35 249

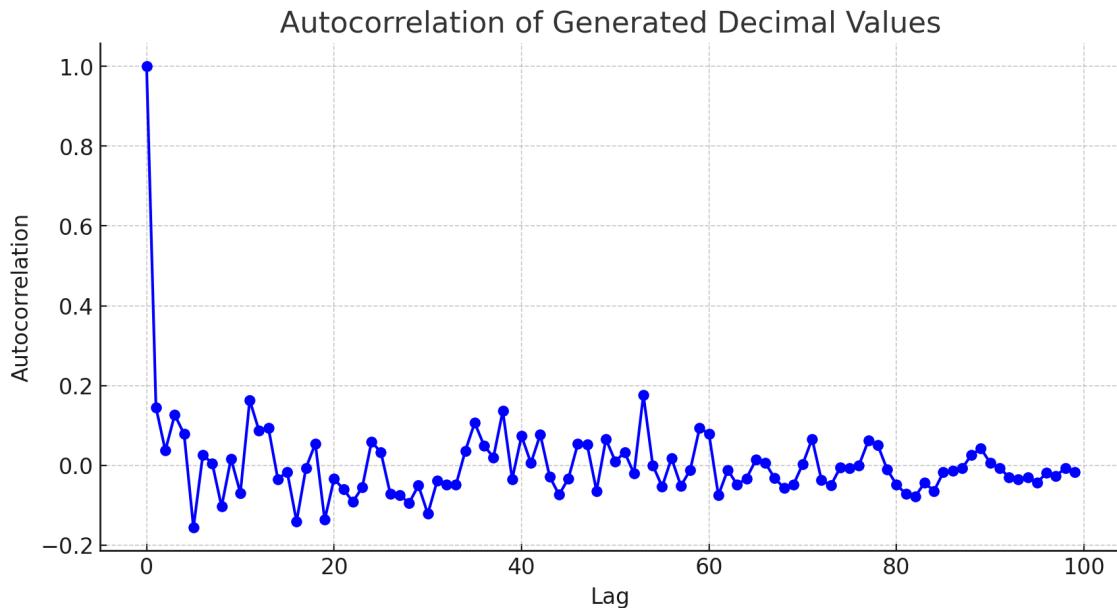
---

209 132 123 53 56 104

---

32 46 163 37

---



## 5.2. Analysis

The detailed analysis of autocorrelation plot provides insight into whether quantum random number generator is working as intended. A successful random number generator should show low autocorrelation values, especially at higher lags, which indicates that the numbers are statistically independent. Any deviations from this pattern, such as consistent peaks or long-term correlations, could suggest that the generator isn't producing truly random numbers, and you may need to revisit the noise source or algorithm.

### Initial Review of the Autocorrelation Plot:

The plot typically displays the autocorrelation values on the y-axis and the lag (the number of steps between data points) on the x-axis. The first few autocorrelation values are especially important because they give us an immediate sense of how the generated data behaves over small time intervals (lags). Interpretation of the First Few Autocorrelations:

For random data, we expect the autocorrelation values to be close to zero at all lags. This would indicate that each value in the series is independent of others, which is a characteristic of truly random sequences. If the autocorrelation is high at lag 1, it suggests that the data points are highly correlated with each other and therefore not random. For the first 10 autocorrelation values, we should look for a trend: If they fluctuate around zero (both positive and negative), it suggests randomness. If they remain high or exhibit patterns, there may be some underlying structure to the data (not truly random). Longer-Lag Auto correlations:

For truly random sequences, we expect the autocorrelation values to drop off quickly as the lag increases. This means that there is no longer any significant relationship between the data points after a certain lag. Decay of autocorrelation: A slow decay might indicate a non-random process or periodicity, suggesting that the data could be influenced by some deterministic process. Peak and Dip Behavior:

Positive peaks in the autocorrelation graph suggest that the data at that lag are cor-

related in a positive way (i.e., similar values appear at that lag). Negative peaks indicate that the values at that lag are inversely related (i.e., opposite values appear). Expected Results for QRNG (Quantum Random Number Generator):

Since you are generating random values based on quantum principles, you would ideally expect a low and fluctuating autocorrelation across all lags. A good quantum random number generator should produce values that exhibit independence over time, resulting in an autocorrelation plot that fluctuates around zero without any significant patterns or periodicity. Interpretation of Results:

If the autocorrelation graph is close to zero for all lags, it is a strong indication that the random number generator is working as expected. This would suggest that the generated numbers are statistically independent, which is a crucial property for randomness. If you observe significant peaks or periodic patterns in the autocorrelation function, it may indicate that the random number generator is biased or not truly random. In this case, further adjustments to the noise source or algorithm might be necessary to improve randomness. Possible Causes for Non-Randomness:

Noise source issues: If the analog noise from the Zener diodes is not properly filtered or is too predictable, it could lead to autocorrelation at certain lags. Algorithmic factors: If there are any biases or patterns in the way bits are extracted from the noise, this could result in non-random behavior in the autocorrelation function.

## 6. Range and Scope of the project

### 6.1. Range

- Hardware Focus:

The project is focused on developing a QRNG using Zener diodes that exploit quantum tunneling phenomena. The hardware design and integration with the Arduino microcontroller and LCD screen will be central to the project.

- Software Control:

The software component will involve programming the Arduino to collect data from the Zener diodes and convert it into binary numbers. The display of the generated numbers on an LCD screen and serial output to monitor the random numbers will be part of the scope.

- Statistical Analysis:

The project will also include the analysis of the generated random numbers through statistical methods, such as calculating the autocorrelation function to assess randomness. Conducting 100 trials of the QRNG to evaluate the statistical properties of the generated data will be part of the project.

- Testing and Validation:

The project will include testing the QRNG's performance by generating numbers and checking if they meet randomness criteria. Autocorrelation and other tests (e.g., entropy analysis) will be conducted to validate the randomness of the generated numbers.

## 6.2. Scope

- Objective:

The primary objective is to design and implement a quantum random number generator using quantum tunneling phenomena in Zener diodes. The project will aim to validate the randomness of the numbers generated by the QRNG by conducting statistical tests, including autocorrelation analysis, over 100 trials.

- Project Deliverables:

A fully functional hardware setup consisting of an Arduino microcontroller, Zener diodes, and an LCD display. A detailed report on the design and implementation of the QRNG, including the source code for the Arduino. A comprehensive analysis of the randomness of the generated numbers using statistical methods (autocorrelation, binary entropy, etc.). A final conclusion on the effectiveness of the QRNG based on the test results.

- Constraints:

The project is limited to the hardware and software integration using an Arduino platform. The focus is on developing a prototype QRNG; full-scale production and commercialization are not part of the scope. The testing will be limited to 100 trials of randomness testing.

- Exclusions:

The project will not involve advanced quantum hardware or quantum computational methods beyond the simple application of quantum tunneling in the Zener diodes. Commercial applications and scalability of the QRNG will not be explored in this project, though the potential for such applications may be discussed.

- Limitations:

The performance of the QRNG will be limited by the noise source and the quality of the Zener diodes. The randomness of the generated numbers is constrained by the experimental setup and the limitations of Arduino's processing power and timing precision.

## 7. Project Extension

### 7.1. Improving Hardware Design:

Higher Quality Components: Upgrade the Zener diodes or use other quantum tunneling-based devices like tunnel diodes to increase the quality of randomness. Noise Amplification: Implement hardware noise amplifiers or filters to improve the quality of the random bits generated by the noise source. Multiple Noise Sources: Add more noise sources to increase entropy and robustness of the generated numbers. This could include using a photodiode and exploiting shot noise, which is another source of quantum randomness. Microcontroller Alternatives: Upgrade from Arduino to more advanced microcontrollers like the Raspberry Pi or FPGA to achieve higher sampling rates or improve data processing.

### 7.2. Improved Software Algorithms:

Post-Processing Algorithms: Implement post-processing algorithms like the Von Neumann extractor to eliminate bias and improve randomness after data generation. Higher Entropy Generation: Use more advanced statistical techniques to ensure the generated

numbers are more uniformly distributed and truly random. Data Visualization: Create more advanced visualization tools to display randomness, such as graphical representations of autocorrelation and entropy over time.

### **7.3. Testing Validation:**

Extended Statistical Testing: Perform more extensive statistical tests like NIST SP800-22, Diehard tests, or the TestU01 suite to validate the randomness of the generated numbers. Real-time Analysis: Implement real-time analysis of randomness and display feedback on the quality of the numbers generated.

### **7.4. Security Enhancements:**

Cryptographic Applications: Investigate how the generated random numbers can be used in cryptographic applications such as key generation for encryption algorithms. This would involve securing the random number generator and ensuring its output meets security standards. Entropy Pool: Build an entropy pool that combines randomness from multiple sources to ensure the random numbers generated meet higher standards of unpredictability, making it suitable for use in secure applications.

### **7.5. Applications:**

Secure Key Generation: Investigate the potential use of your QRNG in cryptographic systems to generate secure keys for encryption and digital signatures. Monte Carlo Simulations: Explore how your QRNG can be applied to Monte Carlo simulations, especially for scientific computing tasks that require a high level of randomness. Gaming and Lottery Systems: Develop a system that uses the QRNG to power random number generation for online gaming or lottery applications, ensuring that they are provably fair and secure.

### **7.6. Integration and Deployment:**

Wireless Communication: Implement wireless communication protocols (e.g., Bluetooth, Wi-Fi) to transmit the random numbers to remote systems. Web Interface: Create a web interface to allow users to access the QRNG system remotely and retrieve random numbers for use in applications. Portable Device: Package the QRNG into a portable device that can be used in field applications like security or scientific experiments.

### **7.7. Scalability and Commercialization:**

Mass Production: Investigate the feasibility of scaling the design for mass production, which could make your QRNG useful in industries requiring high-quality random numbers, such as finance and cryptography. Certification and Standardization: Work towards getting your QRNG certified for compliance with international standards for randomness, such as ISO/IEC 18031 for random number generation.

## **Acknowledgments**

I would like to acknowledge the following sources and websites that have contributed to the completion of this project:

- <https://docs.arduino.cc/hardware/nano/>
- <https://phys.libretexts.org/>
- Zener Theory and Design Considerations Handbook HBD854/D Rev. 1, Dec2017
- OpenAI. (2024). \*ChatGPT\* (Version 4) [Large language model]. OpenAI. <https://chat.openai.com/>
- I would like to sincerely thank my team members, Karthika P and Shardul Ghore, for their contributions and support throughout the project.