ELE201 Semester Project

Thobias Høivik

Contents

1 Introduction **3**

2 Theory **4**

2.1 Randomness and Entropy 4

2.2 True vs. Pseudo Randomness 4

2.3 Statistical Tests of Randomness 4

2.4 Tentative Predictions 5

3 Implementation **6**

3.1 Hardware Setup 6

3.2 Data Transmission 6

3.3 Processing 6

4 Results and Analysis **7**

4.1 Network Design 7

4.2 Subnetting and Addressing 7

4.3 Data Security and Transmission Integrity 7

5 Conclusion **8**

5.1 Future Work 8

1 Introduction

In this project we use the STM32 microcontroller along with analog sensors to gather data that we analyze for statistical and informational randomness, with the goal of evaluating how feasible such a setup is as a hardware-based true random number generator (TRNG). The data will be sent via serial connection to a computer to analyze the quality of the data.

We will combine theory from information theory, probability, signal processing, and networking to assess both the quality of randomness and how such a system could integrate into a distributed architecture.

We will discuss:

- Theoretical background on randomness and entropy.
- Implementation and data transfer.
- Statistical analysis and randomness testing.
- Viability discussion and possible improvements.
- Network design for scaling such systems.

2 Theory

2.1 Randomness and Entropy

Randomness can be defined both in a statistical and algorithmic sense. A sequence is considered random if it is unpredictable and lacks compressible structure.

Definition 2.1: Shannon Entropy

Given a discrete random variable X that takes values in χ with probability distribution $p : \chi \to [0,1]$, its entropy is

$$H(X) = -\sum_{x\in\chi} p(x)\log_2 p(x).$$

Entropy measures the uncertainty of a random variable. For a binary sequence, $H(X) = 1$ indicates perfectly balanced and maximally unpredictable bits.

Intuitively, high entrop should mean unpredictable bits, while low entropy should mean that the bits are structured and predictable. This sounds very good for our purposes, but if we look at a string like "$010101010101\ldots$", where we alternate 0s and 1s, we get a shannon entropy of 1 which is as high as we can go. However, this string of bits very clearly has a predictable structure, which entropy alone cannot take into account.

Definition 2.2: Min-Entropy

The min-entropy of a random variable X is

$$H_{\text{min}}(X) = -\log_2\left(\max_x p(x)\right),$$

representing the worst-case predictability of any outcome.

2.2 True vs. Pseudo Randomness

A pseudo-random number generator (PRNG) produces deterministic sequences from an initial seed, while a true random number generator (TRNG) relies on physical entropy. For hardware-based RNGs, ensuring unbiased and unpredictable output requires:

- High-quality analog entropy sources.
- Proper sampling rates and whitening.
- Statistical post-processing.

2.3 Statistical Tests of Randomness

We use the NIST SP 800-22 statistical test suite as a theoretical foundation. The most relevant tests include:

- Frequency (Monobit) Test
- Runs Test
- Autocorrelation Test

- Approximate Entropy Test
- Linear Complexity Test

Each of these tests returns a p-value indicating how likely the observed sequence could occur under true randomness.

2.4 Tentative Predictions

3 Implementation

3.1 Hardware Setup

We will use the STM32 microcontroller, connected to one or more analog sensors. Candidate sensors include:

- Microphone or sound sensor.
- Light-dependent resistor (LDR).
- Temperature sensor.

The microcontroller samples the analog voltage via the ADC and stores or streams the digital values over a serial or network interface. For now we will be working with a light sensor, sampling at 12 bits and looking at bits in the lower end of the range as they will be most prone to interference and noise.

3.2 Data Transmission

Data is transmitted from the STM32 to a computer for analysis. This can be done using:

- UART/Serial connection.
- Ethernet or WiFi (via ESP module or similar).

For our purpose of analyzing the data for randomness a serial connection will do and we will use UART, sending the data to a computer where we can analyze the data.

3.3 Processing

Our goal is to transform the raw data stream (which is **biased** and **predictable**) into a shorter, statistically perfect random stream.

1. Raw Bit Extraction
   - **Technique:** Only use the least significant bits of the ADC output.
   - The LSBs are dominated by the desired random noise sources and are less influenced by the much larger and more predictable signal (e.g. light level in the case of a light sensor). Thus the MSBs should be discarded entirely as they should exibit fairly low entropy.
2. Entropy Conditioning (Post-Processing) Since the LSBs will more than likely still contain residual bias and correlation we require a powerful **Entropy Extractor** for cryptographic quality.
   - **XOR Folding:** XOR a bit with one a few steps behind. E.g. $\text{Bit}_{\text{Out}} = \text{Bit}_i \oplus \text{Bit}_{i-N}$, where N is chosen to be slightly larger than the observed correlation length.
   - **Cryptographic Hash Extractor (NIST SP 800-90B):** Collect a large buffer of L raw bits. Estimate H_{min} per bit. Take the buffer, and hash it (e.g. SHA-256) to produce a fixed-length output.

     Ths downside with this approach is that we have to sample many times to get a sufficiently large buffer.

4 Results and Analysis

4.1 Network Design

A possible setup:

- Each microcontroller is connected to a local router.
- A dedicated subnet for sensor nodes (e.g., 192.168.10.0/24).
- Central analysis server on a separate subnet.

4.2 Subnetting and Addressing

We show how to subnet the network efficiently:

- Example: dividing a /24 into four /26 subnets.
- Assigning IP ranges for sensors, analysis nodes, and administration.

4.3 Data Security and Transmission Integrity

We briefly discuss:

- Packet integrity verification (CRC or checksum).
- Optional encryption for data in transit.
- Synchronization and time-stamping for accurate sampling.

5 Conclusion

We summarize:

- Theoretical feasibility of analog-sensor-based TRNGs.
- Experimental results and limitations.
- Potential for distributed entropy networks.

5.1 Future Work

- Hardware whitening circuits and amplification.
- FPGA or ASIC implementations.
- Scaling to larger networks and entropy pooling.

References

(Include IEEE papers, arXiv articles, and videos cited earlier.)