Security

Random number generator Ver. 2.0

Robin Pynckels
Filip Pynckels

November 16, 2020
February 20, 2021
(work in progress)

Documents:

http://users.telenet.be/pynckels/2020-4-Random-number-generator_ver_2-0.pdf

Table of contents

List of figures 3
1. Introduction 4
2. Why do we need a hardware random number generator? 6
2.1. What are random numbers? 6
2.2. What are pseudo-random numbers 6
2.3. Why we create a hardware random number generator 7
3. Quantum tunneling (Zener effect) 8
3.1. Wave function, Schrödinger equation, probability density 8
3.1.1. One-dimensional time-dependent system 8
3.1.2. One-dimensional time-independent system 8
3.2. A system with a potential barrier 9
3.2.1. Region 1 - Before the barrier 10
3.2.2. Region 2 - In the barrier 11
3.2.3. Region 3 - After the barrier 12
3.2.4. Boundary conditions 12
3.2.5. Transmission factor 14
3.2.6. Conclusions 15
3.3. Bipolar junctions as entropy generators 16
3.3.1. Zener diode 16
3.3.2. Bipolar junction transistor 17
4. Hardware design 19
4.1. Design choices 19
4.2. Entropy generation 19
4.3. Differential amplification 20
4.4. Voltage boost converting 22
4.5. Digitizing the analog differential signal 23
4.6. Communicating by USB 24
4.7. Regrouping all parts in one schematic 25
4.8. Designing the PCB 26
4.9. The final hardware 26
5. Software design 28
5.1. Device software 28
5.2. Host software 28
5.2.1. Linux 28
5.2.2. Mac OSx 28

5.2.3. Windows 28
6. Quality tests 29
6.1. Hardware quality tests 29
6.2. Software quality tests 29
6.2.1. Device software 29
6.2.2. Host software 29
6.3. Random number quality tests 29
7. Documentation 30
7.1. Hardware 30
7.1.1. Implementation 30
7.1.2. Installation 30
7.1.3. Verification methods 30
7.2. Software 30
7.2.1. Implementation 30
7.2.2. Installation 30
7.2.3. Verification methods 30
8. Conclusion 31
Appendix 1 - Project outline 32
Appendix 2 - Schematic 33
Appendix 3 - Bill of materials 34
Appendix 4 - Optimized differential amplifier 35

List of figures

Figure 1 - Random number generator prototype 4
Figure 2 - A system with a potential barrier 9
Figure 3 - Diode noise 16
Figure 4 - Diode schematic 16
Figure 5 - Transistor schematics 17
Figure 6 - Double entropy generation 20
Figure 7 - Differential signal example 20
Figure 8 - Differential amplification 21
Figure 9 - Boost converter 22
Figure 10 - Random number generator schematic 25
Figure 11 - PCB Front side 26
Figure 12 - PCB Back side 26
Figure 13 - Random number generator front 27
Figure 14 - Random number generator Back 27
Figure 15 - Random number generator front populated 27
Figure 16 - Random number generator Back populated 27

1. Introduction

Since some time, the Direction ICT of the Belgian Ministry of Home Affairs is looking into crypto systems. Within that framework, a preliminary study has been made [1]. With this study as a basis, priorities were decided upon, and a first project outline was made (*Appendix 1*). As can be seen in the project outline, the first priority was the development of a hardware random number generator. Since budgets are limited, outsourcing this project is not an option. Since staff is limited, doing an internal development is not an option neither. So, the authors decided to invest their private time in this project.

The actual state of play is that the first tests of the actual hardware (version 2.00.06) are very satisfying. Pictures of the actual PCB can be found in *Figure 13*, *Figure 14*, *Figure 15* and *Figure 16*. Random numbers are generated at a speed coherent with the USB port that can pass them to the computer/program that needs them. We are sure that internal generation speeds can be upgraded to 50 Gb/s when chosen the right components, but that would only be useful if the generator is used as a hardware component in a broader hardware system, with a means to communicate the random data at 50 Gb/s to that broader system. Also, this would force us to make a PCB that is larger than the one that we wish to create (the size of a small USB-stick, maximum 40mm x 15mm).

This document is a work in progress and must be read as such. It contains an as extensive as possible documentation concerning the hardware random number generator that is being developed. All information in this document and the corresponding archive file is kept under the MIT open software license, extended to the intellectual, design and hardware aspects of the project. This means in major lines that everything from this document, from the hardware design files, from the software that comes with it, from the ideas behind the design, etc. can be copied and used, as long as our intellectual and authoring rights are not denied nor violated, and as long as we are referenced as the original authors/developers/designers.

A prototype was made as a shield for an ATmega328 based business card that we have designed some time ago. This prototype was evaluated on strengths and weaknesses and led to the optimized design described in this document and from which a prototype is shown in Figure 1.

FIGURE 1 - RANDOM NUMBER GENERATOR PROTOTYPE

As mentioned above, it is our goal to be as transparent and complete as possible in the description that follows. The reason is that we are strong believers in the open source/hardware economic model. We estimate that source/hardware must be accompanied with sufficient documentation and that the sources should be open for the public. It is not our goal to keep aspects hidden, but we are not able to estimate everything you want to know about this project,

[1] "*Cryptosystems in an ever-changing world*", Freya Verbeke (under the guidance of Prof. Marnix Van Daele, UGent, Faculty of Sciences, Department of Applied mathematics, computer science and statistics), 8 august 2019

and, as stated before, this document is a work in progress. Hence, if something is not mentioned in this document, do not hesitate to contact us. We are more than willing to explain things further and to help you out with your project.

Below, we elaborate on the design choices we've made. We will elaborate further on some theoretical background, on the hardware functioning and on the software functionalities. Finally, we will go into the testing of the quality of the random numbers generated by this device.

We hope that you enjoy reading this document and that you find the inspiration in it that you are looking for.

2. Why do we need a hardware random number generator?

2.1. What are random numbers?

Random numbers are numbers of an arbitrary length that are generated by a software system, a hardware system or a combination of both, and that are not predictable before they are generated, whatever the number of random numbers the system has already generated.

The way random numbers can be generated is by measuring a physical phenomenon that is known to have a large degree of entropy. Examples of this kinds of phenomena are atmospheric noise, thermal noise, radioactive decay, quantum phenomena, etc. The output of each of these phenomena can be quantized. This quantization must in almost all cases be adjusted to avoid biasing of the measurements. The result are random bits, random bytes, random analog values, etc.

2.2. What are pseudo-random numbers

When a software system is used to generate random numbers, a computational algorithm is used. Such a computational algorithm can generate longs sequences of apparently random numbers. However, it is characteristic for such systems to use an initial value (called a seed) to start generating numbers. It is also characteristic for most such systems to be able to produce a limited sequence of apparent random numbers before starting to regenerate the exact same sequence. The apparent random numbers that are generated by computational algorithms are called pseudo-random numbers.

A potential weakness of pseudo-random number generators is that the entire sequence can be reproduced of one knows (or can guess) the seed value.

A second weakness of pseudo-random number generators is that, if one needs more random numbers than the sequence that can be generated, one gets the same sequence over and over again. On the one side, this makes life of hackers a lot easier, because they can start using the already known sequence. On the other side, if the random numbers are used to test systems (science, engineering, …), one executes the same test repeatedly once the end of the sequence is passed.

Although a pseudo-random number generator only needs a few computer instructions to generate a new pseudo-random number, some physical phenomena permit to generate millions of random bits per second, making most of the pseudo-random number generators relatively slow for high-end applications. On top of that, multiple of the same hardware random generators can be used on the same machine, and they will all generate real independent random numbers. When using the same algorithm two or more times in parallel, one will see appearing the same sequences after some time, meaning that no more than one instance of the software random number generator can reliably be used at the same time on a machine.

Although a pseudo-random number generator is deterministic, it is less expensive than most of the real random number generators (let's call them hardware random number generators), it can be used in any device that has some kind of a programmable controller or processor, and most of the algorithms are publicly available.

Also, most of the pseudo-random algorithms are well documented and transparent enough to be reliable for use in a professional environment as well as in amateur applications. This can hardly be said of most of the existing hardware random number generators, which seem to be closed box, no longer available, no longer maintained, or all the above.

2.3. Why we create a hardware random number generator

On a professional level, strong encryption can only be obtained when there are no conceptual or implementation flaws in the algorithms and when one can use non-predictable random numbers.

On a personal level, we wanted to create a hardware random number generator that is:

- entirely and well documented from the theoretical background of the physical phenomenon to the sources and the installation instructions
- open to the public, meaning that it is open intellectual property, open design, open hardware, open software and distributed under an MIT creative commons license (see titlepage)
- cheap to build and cheap to use
- flexible to use on and with native integration for a wide range of machines and in a wide range of production environments
- reliable by design but especially reliable due to full user verifiability during build and during use.
- verified by several professionals, each in their line of expertise
- continuously maintained and supported.

Due to the above premises, components like Geiger-Müller tubes are not feasible. The system we want to use must use generally available components that are sufficiently documented and that are not intellectual property. So, there was not much choice left but to look at electronic components that permit us to measure their internal quantum phenomena, or at least one internal quantum phenomenon. The most common components that meet this requirement are diodes and transistors. With both components, one can induce a phenomenon of measurable quantum tunneling (also called the Zener effect).

The rest of this paper will be focused on the design and implementation of a hardware random number generator and the accompanying software and documentation based on avalanche noise due to the Zener effect.

3. Quantum tunneling (Zener effect)

In the previous chapter, we dove into some possibilities to generate entropy. The one that appealed the most was quantum tunneling, since it can even be implemented on microscopic scale. In this chapter, we will go deeper into the theory behind quantum tunneling, starting from a few general known premises.

3.1. Wave function, Schrödinger equation, probability density

3.1.1. One-dimensional time-dependent system

In this paper, we will start from the given that the basis of quantum physics is pretty much the fact that a particle can be represented as a wave function. In a time-dependent system, we write:

$$\Psi(x,t) \tag{1}$$

The wave function (1) is the general solution of the time-dependent Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\Psi(x,t) + V(x,t)\Psi(x,t) \quad with\ \hbar = \frac{h}{2\pi}$$

The probability that a particle can be found at a certain place can be expressed by a probability density. The probability in a time-dependent system that a particle with wave function (1) can be found at position x at moment t is

$$P(x,t) = \Psi(x,t)^*\,\Psi(x,t)$$

In the case where the wave function is a real function, this can also be written as

$$P(x,t) = |\Psi(x,t)|^2$$

The probability that a particle with wave function (1) can be found between position x_1 and x_2 at moment t is

$$P(t)_{[x_1,x_2]} = \int_{x=x_1}^{x_2} \Psi(x,t)^*\,\Psi(x,t)\,dx$$

Since P(x,t) is a probability distribution at each moment t, it should be clear that

$$P(t)_{[-\infty,+\infty]} = \int_{x=-\infty}^{+\infty} \Psi(x,t)^*\,\Psi(x,t)\,dx = 1$$

And hence that $\Psi^*(x,t)$ and $\Psi(x,t)$ must converge to 0 at -∞ and at +∞.

3.1.2. One-dimensional time-independent system

In a time-independent system, the wave function of a particle is written as:

$$\Psi(x) \tag{2}$$

The wave function (2) is the general solution of the time-independent Schrödinger equation:

$$E(x)\,\Psi(x) = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\Psi(x) + V(x)\Psi(x) \quad with\ \hbar = \frac{h}{2\pi} \tag{3}$$

The probability density in a time-independent system that a particle with wave function (2) can be found at position x is

$$P(x,t) = \Psi(x)^*\,\Psi(x)$$

In the case where the wave function is a real function, this can also be written as

$$P(x) = |\Psi(x)|^2 \tag{4}$$

The probability that a particle with wave function (6) can be found between position x_1 and x_2 at moment t is

$$P_{[x_1,x_2]} = \int_{x=x_1}^{x_2} \Psi(x)^*\,\Psi(x)\,dx$$

Since P(x) is a probability distribution, it should be clear that

$$P_{[-\infty,+\infty]} = \int_{x=-\infty}^{+\infty} \Psi(x,t)^*\,\Psi(x,t)\,dx = 1$$

And hence that $\Psi^*(x,t)$ and $\Psi(x,t)$ must converge to 0 at -∞ and at +∞.

3.2. A system with a potential barrier

Let us suppose there is a particle moving from the left to the right. It is moving in a system with a potential barrier in the middle. This barrier has a fixed width ω and a potential V_0 (see *Figure 2*). The particle has a total energy E smaller than V_0 since otherwise, we would be describing a free particle that is not hindered by the potential barrier.

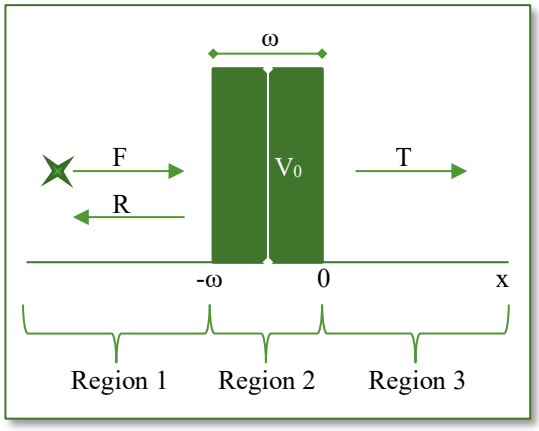


FIGURE 2 - A SYSTEM WITH A POTENTIAL BARRIER

We have used the following symbols in *Figure 2*:

- F : forward wave with particle energy E
- T : Tunneled wave
- R : Reflected wave
- ω : width of the potential barrier
- V_0 : potential energy of the potential barrier

We will be working with the time-independent Schrödinger equation. Due to the different potentials in the regions 1, 2 and 3, we will have to split up our calculations per region. However, the total energy of the particle is constant,

$$E(x) = E \tag{5}$$

and for the three regions goes that:

$$E \leq V_0 \tag{6}$$

Note that on the boundaries between two regions, the wave function and its first derivative must be continuous.

3.2.1. Region 1 - Before the barrier

For this region goes that

$$x \leq -\omega$$

and that

$$\mathrm{V(x)} = 0 \tag{7}$$

Given (3), (5) and (7), the Schrödinger equation for this region is

$$E\,\Psi(x) = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\Psi(x)$$

Or with other words:

$$\frac{d^2}{dx^2}\Psi(x) = -\frac{2mE}{\hbar^2}\,\Psi(x) \tag{8}$$

If we define a new constant κ as

$$\kappa = \sqrt{\frac{2mE}{\hbar^2}}$$

then

$$\kappa^2 = \frac{2mE}{\hbar^2}$$

and hence we derive from (8) that

$$\frac{d^2}{dx^2}\Psi(x) = -\kappa^2\,\Psi(x) \tag{9}$$

The general solution for equation (9) is

$$\boxed{\Psi(x) = \alpha_0 e^{i\kappa x} + \alpha_1 e^{-i\kappa x} \quad with\ \alpha_0\,, \alpha_1\ \in\ \mathbb{R}\ \ and\ with\ x \in [-\infty, -\omega]} \tag{10}$$

α_0 and α_1 are constants that will be calculated further in this paper.

3.2.2. Region 2 - In the barrier

For this region goes that

$$-\omega \leq x \leq 0$$

and that

$$\mathrm{V(x)} = V_0 \tag{11}$$

Given (3), (5) and (11), the Schrödinger equation for this region is

$$E\,\Psi(x) = -\frac{\hbar^2}{2m}\,\frac{d^2}{dx^2}\Psi(x) + V_0\,\Psi(x)$$

Or with other words

$$\frac{d^2}{dx^2}\Psi(x) = \frac{2m(V_0 - E)}{\hbar^2}\,\Psi(x) \tag{12}$$

Due to (6) we know that

$$0 \leq \frac{2m(V_0 - E)}{\hbar^2}$$

This permits us to define a new constant ξ as

$$\xi = \sqrt{\frac{2m(V_0 - E)}{\hbar^2}}$$

So that

$$\xi^2 = \frac{2m(V_0 - E)}{\hbar^2}$$

and so that (12) becomes

$$\frac{d^2}{dx^2}\Psi(x) = \xi^2\,\Psi(x) \tag{13}$$

The general solution for equation (13) is

$$\Psi(x) = \alpha_2 e^{\xi x} + \alpha_3 e^{-\xi x} \qquad with\ \alpha_2\ , \alpha_3\ \in\ \mathbb{R}\ \ and\ with\ x \in [-\omega, 0] \tag{14}$$

α_2 and α_3 are constants that will be calculated further in this paper.

3.2.3. Region 3 - After the barrier

For this region goes that

$$0 \leq x$$

and that

$$V(x) = 0$$

An identical reasoning as for region 1 with a same substitution for the constant κ teaches us that the Schrödinger equation for this region can be written as (9) and that it has the following general solution

$$\boxed{\Psi(x) = \alpha_4 e^{i\kappa x} + \alpha_5 e^{-i\kappa x} \qquad with\ \alpha_4\ , \alpha_5\ \in\ \mathbb{R}\ \ and\ with\ x \in [0, +\infty]} \tag{15}$$

α_4 and α_5 are constants that will be calculated further in this paper.

3.2.4. Boundary conditions

A first condition that we must consider when deciding upon the values of the constants α_i is that there are no boundaries in region 3. This has as a consequence that a particle in region 3, should it ever get there, has an initial direction from left to right (coming from the left of region 3) and cannot bounce back due to a lack of further potential barriers. The chance that a particle suddenly turns back without potential barrier is so small that it can be seen as inexistent. The only way to guarantee that there is no backward movement in region 3 is to set α_5 to zero:

$$\boxed{\alpha_5 = 0} \tag{16}$$

As already mentioned, a wave function and its derivative must be continuous over its entire domain. This is also true at the edges of the potential barrier.

Since we have three regions, there are two edges: one between region 1 and region 2 where x is equal to -ω, and one between region2 and region 3 where x is equal to 0.

Starting from (10) and (14), we can state that

$$\Psi(-\omega) = \alpha_0 e^{-i\kappa\omega} + \alpha_1 e^{i\kappa\omega} = \alpha_2 e^{-\xi\omega} + \alpha_3 e^{\xi\omega}$$

$$\frac{d}{dx}\Psi(x)\Big|_{x=-\omega} = i\kappa\alpha_0 e^{-i\kappa\omega} - i\kappa\alpha_1 e^{i\kappa\omega} = \xi\alpha_2 e^{-\xi\omega} - \xi\alpha_3 e^{\xi\omega}$$

Hence, we find that

$$\alpha_0 e^{-i\kappa\omega} + \alpha_1 e^{i\kappa\omega} = \alpha_2 e^{-\xi\omega} + \alpha_3 e^{\xi\omega}$$

$$\alpha_0 e^{-i\kappa\omega} - \alpha_1 e^{i\kappa\omega} = \frac{\xi}{i\kappa}\alpha_2 e^{-\xi\omega} - \frac{\xi}{i\kappa}\alpha_3 e^{\xi\omega}$$

and finally, that

$$\alpha_0 = \frac{1}{2}\left(1 + \frac{\xi}{i\kappa}\right)\alpha_2 e^{(i\kappa-\xi)\omega} + \frac{1}{2}\left(1 - \frac{\xi}{i\kappa}\right)\alpha_3 e^{(i\kappa+\xi)\omega} \quad (17)$$

$$\alpha_1 = \frac{1}{2}\left(1 - \frac{\xi}{i\kappa}\right)\alpha_2 e^{(-i\kappa-\xi)\omega} + \frac{1}{2}\left(1 + \frac{\xi}{i\kappa}\right)\alpha_3 e^{(-i\kappa+\xi)\omega} \quad (18)$$

Starting from (14), (15) and (16), we can deduce that

$$\Psi(0) = \alpha_2 + \alpha_3 = \alpha_4$$

$$\frac{d}{dx}\Psi(x)\Big|_{x=0} = \xi\alpha_2 - \xi\alpha_3 = i\kappa\alpha_4$$

Hence, we find that

$$\alpha_2 + \alpha_3 = \alpha_4$$

$$\alpha_2 - \alpha_3 = \frac{i\kappa}{\xi}\alpha_4$$

and finally, that

$$\boxed{\alpha_2 = \frac{1}{2}\left(1 + \frac{i\kappa}{\xi}\right)\alpha_4} \quad (19)$$

$$\boxed{\alpha_3 = \frac{1}{2}\left(1 - \frac{i\kappa}{\xi}\right)\alpha_4} \quad (20)$$

Substituting (19) and (20) into (17) and (18) leads to

$$\boxed{\alpha_0 = \frac{1}{4i\kappa\xi}\left((i\kappa+\xi)(i\kappa+\xi)\,e^{(i\kappa-\xi)\omega} + (i\kappa-\xi)(-i\kappa+\xi)\,e^{(i\kappa+\xi)\omega}\right)\alpha_4} \quad (21)$$

$$\boxed{\alpha_1 = \frac{1}{4i\kappa\xi}\left((i\kappa-\xi)(i\kappa+\xi)e^{(-i\kappa-\xi)\omega} + (i\kappa+\xi)(-i\kappa+\xi)e^{(-i\kappa+\xi)\omega}\right)\alpha_4} \quad (22)$$

Summarizing the previous results shows us that we can now substitute α_0 , α_1 , α_2 , α_3 with an expression of α_4 and that α_5 is 0. Considering the expressions (10), (14) and (15), we can see that $\Psi(x)$ is almost known apart from the constant α_4.

However, we still have one condition that we have not yet used:

$$P_{[-\infty,+\infty]} = \int_{x=-\infty}^{+\infty} \Psi(x)^* \, \Psi(x) \, dx = 1$$

which tells us that Ψ*(x) and Ψ(x) must converge to 0 at -∞ and at +∞, and hence that the integral of Ψ*(x) and Ψ(x) will be a real number (not -∞ nor +∞). It has become a habit to use

$$\int_{x=-\infty}^{+\infty} \Psi(x) = 1$$

as a condition. Or in the case of the particle in a system with a potential barrier

$$\int_{x=-\infty}^{-\omega} \Psi(x) + \int_{x=-\omega}^{0} \Psi(x) + \int_{x=0}^{+\infty} \Psi(x) = 1 \qquad (23)$$

Combining expressions (10), (14), (15), and (23) gives us an expression from which α_4 can be deduced. For this paper, however, we need to take a different road.

3.2.5. Transmission factor

In this paper, we focus on entropy. In this part of the paper, we are looking into the entropy generation by means of the use of quantum tunneling. This means that we want to know which ratio of the particles will, on average, tunnel through the potential barrier. Each particle is characterized by its energy level E.

Let us define T(E) as the transmission factor in function of the energy of the particle, being the chance that a particle gets through the barrier, or an average ratio of the number of particles that get through the barrier and the total number of particles.

Expression (10) shows that $|\alpha_0|^2$ is a measure of the total number of particles (left to right traveling wave in region 1) and that $|\alpha_1|^2$ is a measure of the number of particles bounced back on the potential barrier (right to left traveling wave in region 1).

Expression (15) combined with expression (16) shows that $|\alpha_4|^2$ is a measure of the number of particles (left to right traveling wave in region 3) that got through the potential barrier.

$$T(E) = \frac{\alpha_4^* \, \alpha_4}{\alpha_0^* \, \alpha_0} = \frac{\alpha_4^*}{\alpha_0^*} \frac{\alpha_4}{\alpha_0} = \left(\frac{\alpha_4}{\alpha_0}\right)^* \frac{\alpha_4}{\alpha_0} \qquad (24)$$

Note that there is also a definition for a reflection factor R(E).

$$R(E) = \frac{\alpha_1^* \, \alpha_1}{\alpha_0^* \, \alpha_0} = \frac{\alpha_1^*}{\alpha_0^*} \frac{\alpha_1}{\alpha_0} = \left(\frac{\alpha_1}{\alpha_0}\right)^* \frac{\alpha_1}{\alpha_0}$$

T(E) and R(E) must comply to the following equation if we do not want to lose particles:

$$T(E) + \mathrm{R(E)} = 1$$

Instead of taking the difficult road (calculating how much α_4 is based on expression (23)), we will deduce the ratio T(E) from equation (21).

Note that both α_0 and α_4 are dependent on the energy (E) and on the mass (m) of the particle since κ and ξ are dependent on the energy and on the mass of the particle. In theory, we can convert all references to m to references to E, but later on, it will be useful to have the mass dependency explicitly mentioned in the formula for T(E).

Simplifying (21) gives

$$\alpha_0\, 4i\kappa\xi\, e^{-i\kappa\omega} = \alpha_4\left((\xi + i\kappa)^2\, e^{-\xi\omega} - (\xi - i\kappa)^2\, e^{\xi\omega}\right)$$

which leads to

$$\alpha_0\, 4i\kappa\xi\, e^{-i\kappa\omega} = \alpha_4\, \left((\kappa^2 - \xi^2)\left(e^{\xi\omega} - e^{-\xi\omega}\right) + i\, 2\xi\kappa\left(e^{\xi\omega} + e^{-\xi\omega}\right)\right)$$

and to

$$\alpha_0\, 2i\kappa\xi\, e^{-i\kappa\omega} = \alpha_4\, \left((\kappa^2 - \xi^2)\sinh(\xi\omega) + i\, 2\xi\kappa\cosh(\xi\omega)\right)$$

so that

$$\frac{\alpha_4}{\alpha_0} = \frac{\mathrm{i}\, 2\kappa\xi\, e^{-i\kappa\omega}}{\left((\kappa^2-\xi^2)\sinh(\xi\omega)+i\, 2\xi\kappa\cosh(\xi\omega)\right)} \tag{25}$$

Taking (25) and its complex conjugate, and substituting these in (24) and gives us

$$T(E) = \frac{1}{1+\frac{\left(\kappa^2+\xi^2\right)^2 \sinh^2(\xi\omega)}{4\kappa^2\xi^2}}$$

Considering the definitions of κ and ξ leads to the following formula for the transmission factor

$$\boxed{T(E) = \frac{1}{1+\frac{V_0^2}{4\,E\,(V_0-E)}\sinh^2\left(\sqrt{\frac{2m(V_0-E)}{\hbar^2}}\,\omega\right)} \qquad with\ E < V_0\ ,\ 0 < \omega\ and\ \hbar = \frac{\mathrm{h}}{2\pi}} \tag{26}$$

The same kind of logic leads to the calculation of R(E), which is of no use to us within the context of this paper.

3.2.6. Conclusions

When a particle is present in a system with a potential barrier, there are several elements that influence the possibility for the particle to tunnel through the barrier.

- The wider the barrier, the smaller the transmission factor and the less particles that will tunnel through the barrier
- The bigger the mass of the particle, the less chance that the particle will tunnel through the barrier
- The larger the potential of the barrier, the less chance there will be quantum tunneling
- The smaller the energy of the particle, the less chance there will be quantum tunneling

Now that the theoretical base is treated, we will need to find a practical system that provides us with a barrier that makes quantum tunneling feasible, and with a possibility to introduce particles with a certain energy so to make it possible to tunnel through the barrier.

3.3. Bipolar junctions as entropy generators

3.3.1. Zener diode

A diode is a PN bipolar junction device and hence suitable to generate entropy, or avalanche noise as it is also called within this context. There is, however, one element that needs clarification. When measuring the noise generated by a PN junction, we see a sawtooth-ish wave (see *Figure 3*). So, what is going on here?

FIGURE 3 - DIODE NOISE

The schematic to generate avalanche noise with a diode (PN junction) is shown in *Figure 4* on the left. However, a diode has, as most electronic components, also a resistive and a capacitive characteristic. When drawing the capacitive characteristic explicitly we the schematic of *Figure 4* on the right.

C is the junction capacitance of diode D, plus any external capacitance (leads, PCB, etc). Some of the current from R leaks through D, but the rest charges C. Once the voltage reaches a certain level, avalanche breakdown occurs and current flows from C until the avalanche stops. Then the current begins charging C again, which explains the sawtooth measurements comparable with the ones in *Figure 3*.

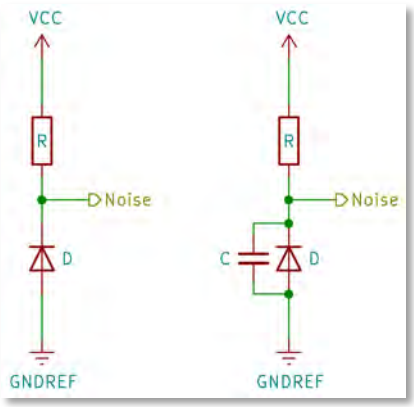


FIGURE 4 - DIODE SCHEMATIC

3.3.2. Bipolar junction transistor

When looking for a system with a potential barrier, bipolar junction transistors seem to do a fairly good job. In fact, one uses only the PN junction between the base and the emitter. One could call it a diode.

NPN bipolar junction transistors are devices that make electrons move from one potential region to another. On the other side, PNP bipolar junction transistors can be considered as devices that make the positive holes move from one potential region to the other.

It would lead to far to treat the entire theory in this paper, but all the equations work just as well with holes in PNPs as they do for electrons in NPNs. You can do the calculations and determine the effective mass of holes and the mobility of holes. The effective mass of holes is larger than that of electrons, and in Si holes are about 2.5 times slower than electrons. Therefore, when using PNP transistors, we need particles with a substantial higher energy than when we use NPN transistors, and they will still move slower. We need an entropy generator (avalanche noise generator) that is fast and consumes as little energy as possible, so the obvious choice is an NPN transistor.

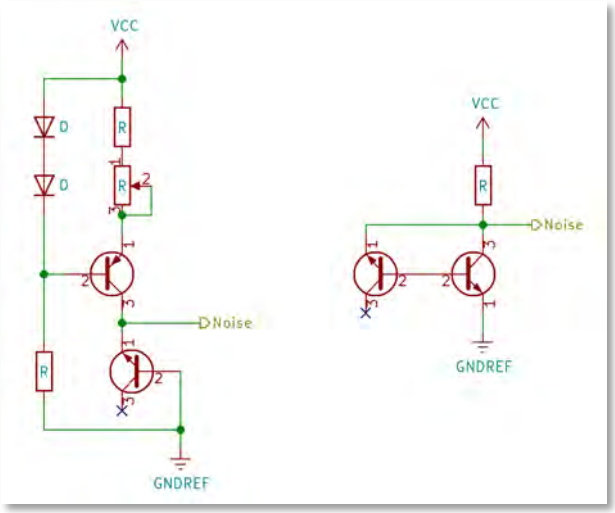


FIGURE 5 - TRANSISTOR SCHEMATICS

So, let us investigate the characteristics of NPN bipolar junction transistors.

- They provide two regions with low potential (Negative region) and one region in the middle with a higher potential (Positive region)
- The barrier is neither too high nor too wide to be insurmountable, and is different per type of BJT transistor, so that a certain liberty of choice exists
- The higher the voltage difference that is created between the emitter and the base of the transistor, the more energy the electrons will have to tunnel from the emitter to the base of the transistor

- Although the tunneling electrons provide a weak current, this current (avalanche noise) is large enough to be amplified and used by an electronic device with plain vanilla components.

It goes without saying that some transistors are better suited to be used as a potential barrier device than others. Experiments have shown that a commonly used NPN bipolar junction transistors transistor, the 2N3904, is suited for our purpose since there is a sufficiently measurable Zener effect between its emitter and its base. So, this is the transistor that will be used for generating entropy and by extension for realizing this project. Two schematics (options) are shown in *Figure 5* at the left and at the right.

Note that in both options of *Figure 5* only one transistor (NPN) is responsible for generating the avalanche noise. The other transistor (a PNP at the left option and an NPN at the right option) is there to amplify the noise signal a first time.

Since the right schematic contains only 3 components, and the left schematic needs 7 components to do roughly the same, we will use the right schematic to reduce the cost, the complexity, and the PCB surface.

4. Hardware design

4.1. Design choices

For reasons of flexibility and usability, we opt for the following choices for the device under development:

- USB connected to a host computer
- Not larger than a normal USB stick
- Contains only plain vanilla components
- Must be reprogrammable to be verifiable by its user.

Since total cost is an important factor, we add the following choices to the requisite list:

- Smallest possible PCB and hence smallest possible components
- 2-layer PCB
- Cheap components
- As few as possible components.
- As few as possible different components.

Since ease of use is an element that counts, we add the following choices to the requisite list:

- As bias independent as possible (prevent or correct power fluctuations, temperature fluctuations, external electro-magnetic fluctuations, etc.)

Since ease of development and price of development tools is an important factor, we add the following choices to the requisite list:

- Only use open hardware / open software for developing
- All development tools (hardware, software) used during development must be easy to obtain for free or for a limited price

4.2. Entropy generation

Based upon the circuit shown in *Figure 5* we constructed a similar transistor entropy generator, but with a low-pass filter between the entropy generating transistor and the power source to filter out most of the influencing power ripple around the frequency that the entropy is generated. To make our entropy generation independent of temperature, low frequency power fluctuations, etc. we need two entropy generators with transistors that are as similar as possible. Transistors from the same construction batch seem a must in this case.

Nota that the Vcc should be higher than 12V (depending on the kind of transistors that are used. We use 2N3904 NPN transistors and 2N3906 PNP transistors. These accept voltages up to 40V so that choosing 15V as Vcc is within the margins of operation.

The circuit that will be used for entropy generation is shown in *Figure 6*. The values for R and C must be calculated to suit the purpose, and to filter out the right frequencies. Keep in mind

that the goal is to have as few as possible different components on the bill of materials. This influences the choice of the R and C values.

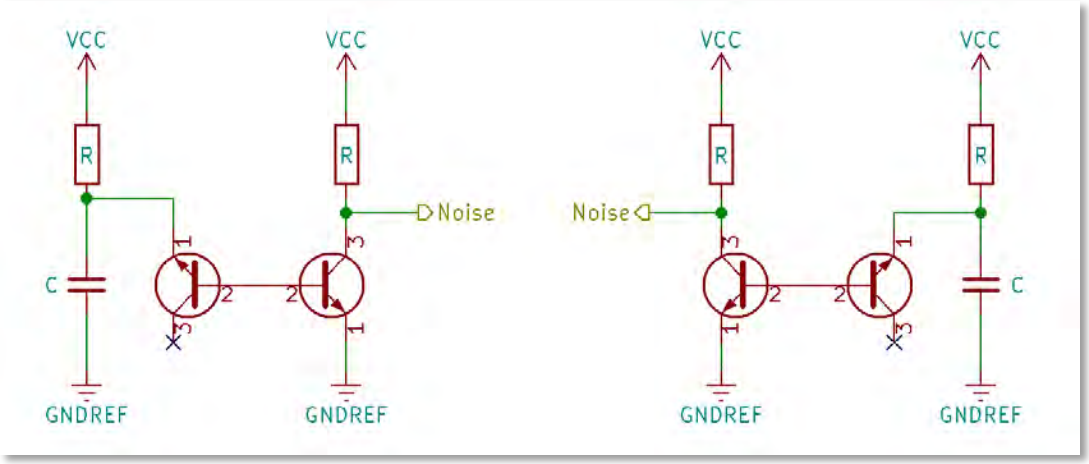


FIGURE 6 - DOUBLE ENTROPY GENERATION

4.3. Differential amplification

When generating noise with electronic components, there are several external influences that should be avoided. The first is electro-magnetic radiation. This influence can be reduced by shielding the USB device with a metallic enclosure.

The second influence is the temperature. As temperature rises or drops, transistors start behaving slightly (or substantially) different. But similar transistors change their behavior in a comparable way. The more similar they are, the more similar their behavioral change is with temperature. So, using two entropy generating circuits that are placed close to each other on a PCB gives us two noise sources with similar distortion.

A third important influence is the rippling of the power supply. The ripples are as well low frequent as higher frequent. Especially the ripples around the frequency of the entropy noise can distort things in a very awkward way.

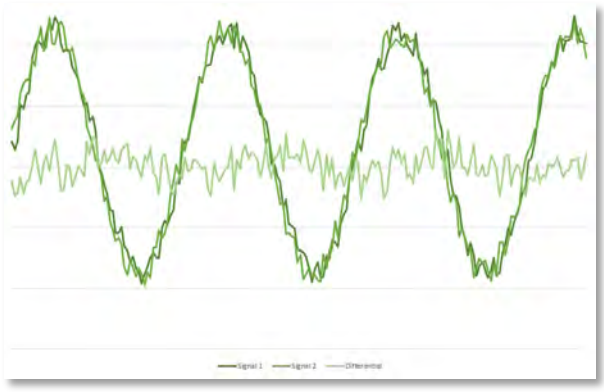


FIGURE 7 - DIFFERENTIAL SIGNAL EXAMPLE

The solution for the mentioned problems (besides shielding the device) is using differential amplification. The entropy signal is too small not to be amplified, so an amplification phase (or multiple amplification stages) is a strict necessity. Looking at the partly amplified signals shows that the external influences are remarkably similar on both the signals (as well in the low frequency domain as in the higher frequency domain). The part of the respective signals coming from the entropy generation however will be different, since they depend on quantum phenomena. So, subtracting the signal coming from one noise source from the signal coming from the second noise source gives us the difference between the noise signals and eliminates the part of the signals coming from the external influence.

Figure 7 shows a theoretical example of what is explained in the previous paragraph. We took two sine graphs (as representation of the external influence). One of the sine graphs has a slight displacement compared to the other one to simulate a slight difference in the external influence on both noise sources. 25% noise was added to the sine graphs giving us "Signal1" and "Signal2" of *Figure 7*. When subtracting "Signal1" from "Signal2", we get the "Difference" signal that is clearly less affected by the mutual influence than both the independent noise sources. The difference signal is almost a pure random signal. It is different from the randomness of the respective signals 1 and 2, but still almost entirely random.

An operational amplifier could do the trick, but since we want as much "verifiability" as possible, we opt to use only base components in our circuit. Besides that element, more components are needed to prepare the signals for use by an operational amplifier (capacitors and voltage dividers) and to make the operational amplifier do its job (at least 4 resistors) than the number of components that are needed to implement the current source and the differential amplifier with transistors. The circuit that we will use is shown in *Figure 8*.

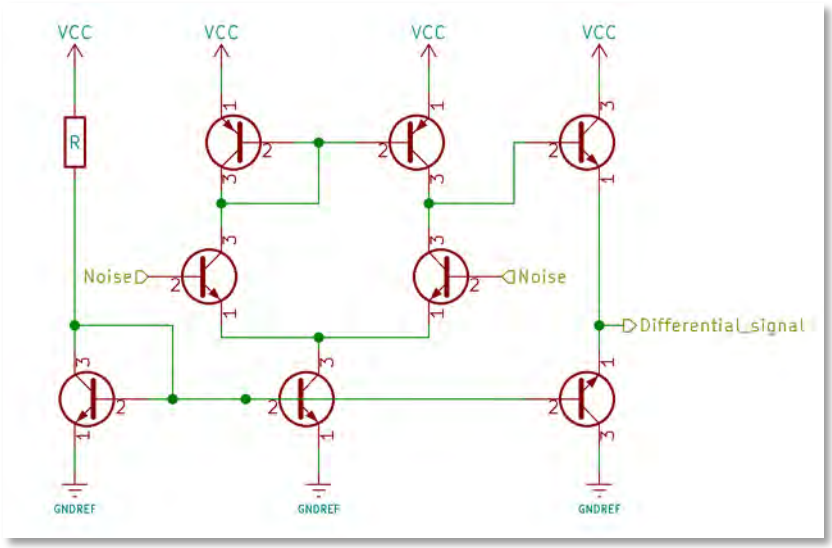


FIGURE 8 - DIFFERENTIAL AMPLIFICATION

It should be clear that the circuit in *Figure 8* will not win the price of the most beautiful baby, but it does what we need it to do with few components: taking the difference between two incoming signals and sufficiently amplifying the result so that it can be read by a small controller on the device.

Just for the readers information, we have joined a more extended operational amplifier design at the end of this document (*Appendix 4*).

4.4. Voltage boost converting

Taking into account the conclusions in *3.2.6* and the characteristics of an NPN transistor in *3.3.2* the Vcc in *Figure 4*, *Figure 5*, *Figure 6* and *Figure 8* has to be high enough to make the quantum tunneling happen.

However, the design choices in *4.1* stipulate clearly that we need a device that is connected to a host machine by means of an USB connection. Knowing that our device will be the most useful when it meets the conditions of an USB low-power device, we only dispose of a theoretical 5V and 100mA, which may drop to 4.4V and 100mA following the technical specifications.

5V is not sufficient to make the tunneling effect in a transistor happen, not to mention 4.4V. So, on the one side, we will need to monitor that the non-entropy-generation part of our circuit can cope with voltage fluctuations between 5.5V and 4.0V (with some margin added), that our entire circuit does not use more than something like 80mA. On the other side, we will need a solution that provides us with at least a constant 15V with as less ripple as possible, and from which we can pull at least 200µA. In reality, these requisites are too broad, and our solution will have to do better than the prerequisites to be on the safe side.

A general solution to boost up a voltage level (and reducing the maximal output current since the power of the system stays the same) is a boost converter running in continuous mode. A typical circuit is shown in *Figure 9*. We will use a block wave, but other wave forms can be considered. However, the calculations below are proper to the use of a block wave, and it is easy to generate a block wave with a controller.

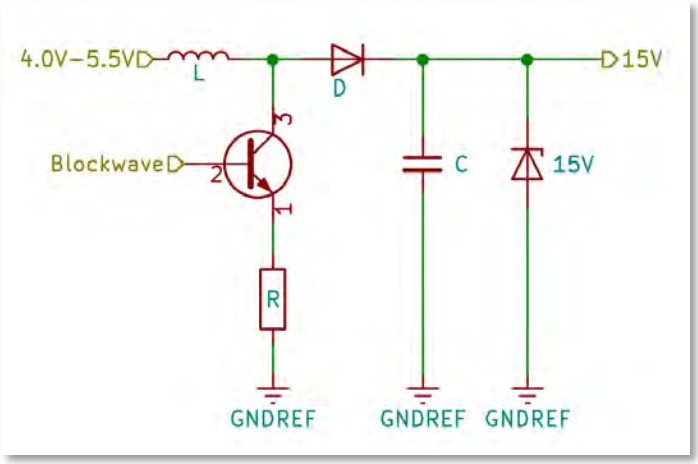


FIGURE 9 - BOOST CONVERTER

When $V_{i,min}$ is the minimum input voltage of the voltage range (4.0V to 5.5V), D the duty cycle of the block wave, and $\langle V_o \rangle$ the desired average output voltage, then the following equation is valid:

$$D = 1 - \frac{V_{i,min}}{<V_o>}$$

Since the circuit in *Figure 9* uses a Zener diode to clip the upper limit of the voltage $<V_o>$, it is advised to take a $<V_o>$ that is little bigger than 15V to make the calculations. For V_i on the other hand, it is advised to use the minimal value $V_{i,min}$ for the calculations so to be sure that the smallest input voltage results in the correct desired average output voltage.

When L_{crit} is the critical inductor value above which the boost converter goes into continuous conduction mode, f is the frequency of the block wave and R is the load resistance (real part of the load input impedance), then:

$$L_{crit} = \frac{V_i^2 \, D \, R}{2 \, f <V_o>^2}$$

Finally, when C_{min} is the minimal capacitance of the capacitor at the output value, and ΔV_o is the maximal acceptable output voltage ripple, then:

$$C_{min} = \frac{<V_o> D}{R \; f \; \Delta V_o}$$

As can be seen from the equations above, the inductor value and the capacitor value depend on the frequency of the block wave, but the ratio of the input and the output voltages only depends on the duty cycle of the block wave.

The only remaining question is how the block wave will be generated. As will be explained later in this paper, we need a small controller to help with certain tasks that are a little complex for a few single components. Since a controller will be used anyway, the same controller can be used to generate an acceptable block wave with adjustable frequency and adjustable duty cycle. The method to create such a block wave is called pulse width modulation (PWM) and a controller should be chosen that has at least one PWM capable output pin.

It goes without saying that theory is just theory and that the efficiency of the components and their placement on the PCB play a role in the final choice of the component values. There are equations that do not consider components as ideal (as the one above) but take into account the mentioned efficiency of the boost converter. However, one has to guess the efficiency (certainly at the first calculation) which means that one can as well guess the efficiency to be 80% as 100% (ideal boost converter). Above, we did the latter.

4.5. Digitizing the analog differential signal

Digitizing analog signals is usually done by means of analog to digital converters that are integrated in one package (ADC-IC), or that are grouped together with other functionality in for instance a micro-controller. During this development, a micro controller can be chosen that contains at least one analog to digital convertor pin.

Such a controller can read an analog input value (between ground reference and its operating voltage) on one of its pins and convert it into a digital value between 0 and a maximal value (e.g. 1024). In most cases, the maximal value is large enough to even read voltage differences of 10mV with a certain confidence. This in turn will permit to program the controller to interpret

values below a certain threshold as representing the binary value 0, and values equal to or above a certain threshold as representing the binary value 1. An algorithm can be developed to assess and guarantee that there are on average as much 0 values as there are 1 values read. This can be done by adapting the threshold and by reassessing in a feedback loop.

Later in this paper, a more extensive description will be given of several methods to assess if the level of the threshold is correct to use the 0 and 1 values to form digital random data. For now, however, it is sufficient to know that a controller is the only thing that is needed to digitize signals coming from the differential amplifier. Eventually a simple voltage divider will be needed also to scale the signal to voltage levels that are acceptable for the controller.

Regarding a first choice of a controller: taking into account the design choices in *4.1* and that the controller must be able to generate block waves and have ADC (analog to digital conversion) functionality orients us in a certain direction (especially the element that we will use generally available components). The most general available and used controllers are undoubtedly Atmel controllers of the ATmega range, Microchip Technology's PIC (peripheral interface controller) controllers and STMicroelectronics STM range. To our knowledge, the ATmega's are the most generally available and used, since they are also part of e.g. Arduino boards. A first choice could be the ATmega328p in a TQFP32 package. However, there are more requirements to come that may change this choice.

4.6. Communicating by USB

The design choices in *4.1* stipulate that the device under development must be connected to a host system by means of an USB port. Communicating through an USB port requires us to also use the USB protocol, which is not all that easy to do starting from scratch. Finding a controller that already incorporates the low level USB functionality is without a shadow of a doubt the best way to go, which excludes the ATmega328p.

There are several different USB device types that can be implemented. Those that come to mind are:

- ADC - Audio Device Class
- CDC - Communications Device Class
- Composite Device Class
- HID - Human Interface Device
- MSC - Mass Storage Class

The CDC device class implements a virtual serial port over USB, which is exactly what is needed in this context. It would lead to far to describe the other device classes, but more information can be found all over the Internet.

In the range of the Atmel controllers, there is the ATmega16U2 controller (that is also used on e.g. the Arduino UNO board) that contains low-level USB functionality. It also has ADC and the PWM functionality on board and seems a good final choice. This controller exists in the small TQFP32 package format, which is suitable for our purpose. On the one side, it is small enough, on the other side, it is easier to hand solder than e.g. the QFN32 package format.

4.7. Regrouping all parts in one schematic

Up until now, we have made several design choices, and we have described the generation of entropy, how to make the circuit less sensitive to fluctuations in the environment, how to generate enough potential difference to permit quantum tunneling in a transistor, how to digitize the resulting differential signal and how we count communicating through an USB port.

Now, the time is ripe to combine all the above ideas and partial circuits into one schematic and have a first overview of the functioning of the entire device under development. The schematic is shown in *Figure 10* and larger in *Appendix 2*.

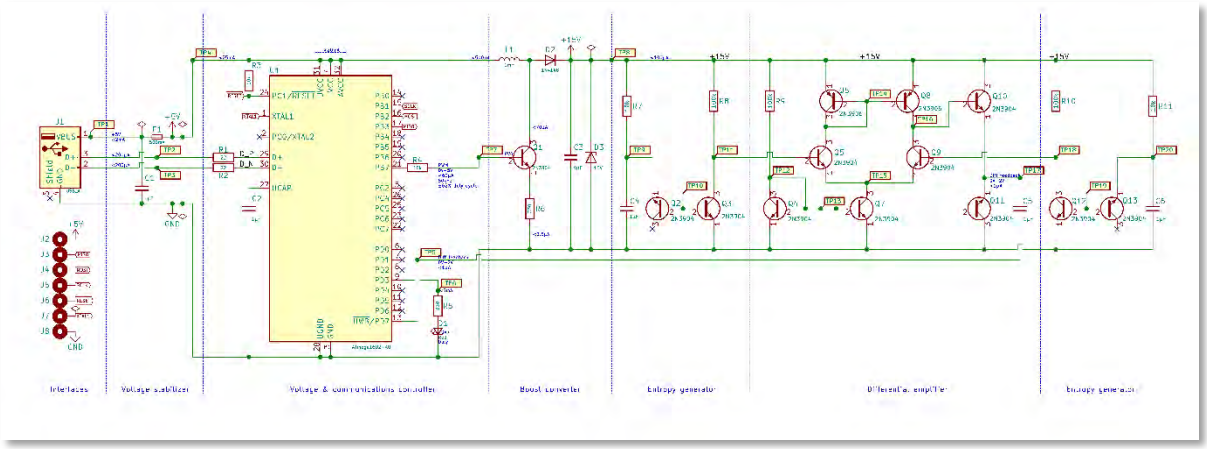


FIGURE 10 - RANDOM NUMBER GENERATOR SCHEMATIC

Nota that a number of test points have been added in the schematic in *Figure 10*. These test points will not be implemented on the PCB as independent components (solder pads) but are put in the schematic to show the reader where he can measure signals to control the functioning of the hardware. Later in this paper, we will add images of the different signal waveforms that can be measured at each test point. We will also mark the place of each test point on a picture of the final PCB, so to make testing the hardware easier.

Note also that we have added 7 interface connection points for the controller. In theory, the controller will be pre-programmed using a different socket before soldering it onto the PCB. We will provide a method to reprogram the controller through the USB port when having local access to the device (by connecting ground with the reset pin for a fraction of a second). The need to have local access to the device makes it impossible to reprogram the device from a remote location through a hacked host device. Since it is our goal to make the hardware as open as possible (and still safe), we need to provide a low-level programming possibility for the controller. This can be done by means of the 7 interface points under the condition to have local access to the device. Again, local access is a requisite to program the device do to prevent remote reprogramming by means of a hacked host device.

4.8. Designing the PCB

When designing the PCB, the choices described in *4.1* play a role. But we also try to make the device as small as possible. We also want to assure that when the device is used in an USB port close to other USB ports, other devices can still be branched into the other ports (read: the device under development can only be as wide as an USB port on an average host system). There is of course also the proximity of certain components to other components that must be guaranteed. But apart from the above, some sort of a structured layout would be nice also. It is our goal to pour some of the final devices in epoxy to show that protection against reprogramming is be possible and cute at the same time. With, of course, the emphasis on the protection against remote and local abuse of the device.

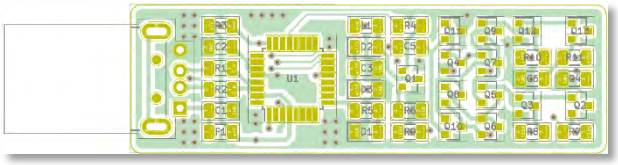


FIGURE 11 - PCB FRONT SIDE

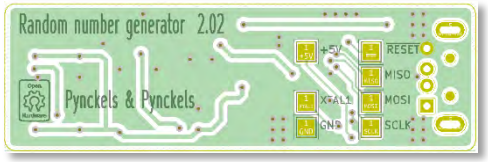


FIGURE 12 - PCB BACK SIDE

Figure 11 and *Figure 12* show the design of the front and the back of the PCB and are as close as possible to the real size of the device under construction. Putting components only on the front of the device was an explicit layout choice, as was putting the breakout (interface connection) solder pads on the back of the device.

Stitching vias have been placed where it was possible. Both the front and the back plane are filled and connected to ground reference. The zones between transistors Q2 and Q3, respectively Q12 and Q13 are not filled to prevent influence of local fluctuations between the front and the back copper layer on the transistor tunneling effect of transistors Q2 and Q13. The backside of these zones has been filled to shield the front components from environmental radiation effects.

4.9. The final hardware

The following images are the PCB version 2.00.06. Their size (unpopulated) is: 37mm x 15mm. This hardware is, at the moment, not entirely tested (hence the test wire on the populated version of the images). But since we want to be as transparent as possible, we already publish this version.

Schematics and PCB designs are available on demand. When tests are conclusive, these files will be put on the website.

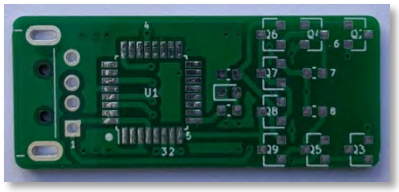


FIGURE 13 - RANDOM NUMBER GENERATOR FRONT

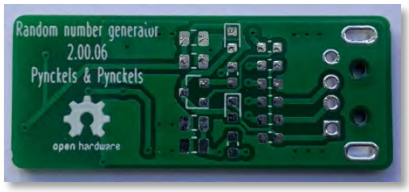


FIGURE 14 - RANDOM NUMBER GENERATOR BACK

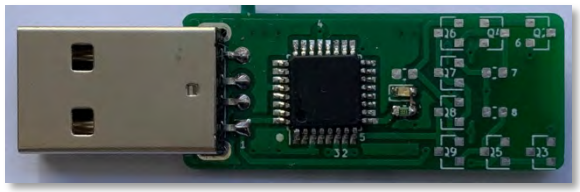


FIGURE 15 - RANDOM NUMBER GENERATOR FRONT POPULATED

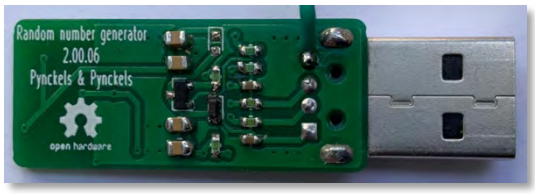


FIGURE 16 - RANDOM NUMBER GENERATOR BACK POPULATED

5. Software design

5.1. Device software

To be documented.

5.2. Host software

5.2.1. Linux

To be documented.

5.2.2. Mac OSx

To be documented.

5.2.3. Windows

To be documented.

6. Quality tests

6.1. Hardware quality tests

To be documented.

6.2. Software quality tests

6.2.1. Device software

To be documented.

6.2.2. Host software

To be documented.

6.3. Random number quality tests

To be documented.

7. Documentation

7.1. Hardware

7.1.1. Implementation

Partially written above.

7.1.2. Installation

To be written.

7.1.3. Verification methods

To be written.

7.2. Software

7.2.1. Implementation

To be written.

7.2.2. Installation

To be written.

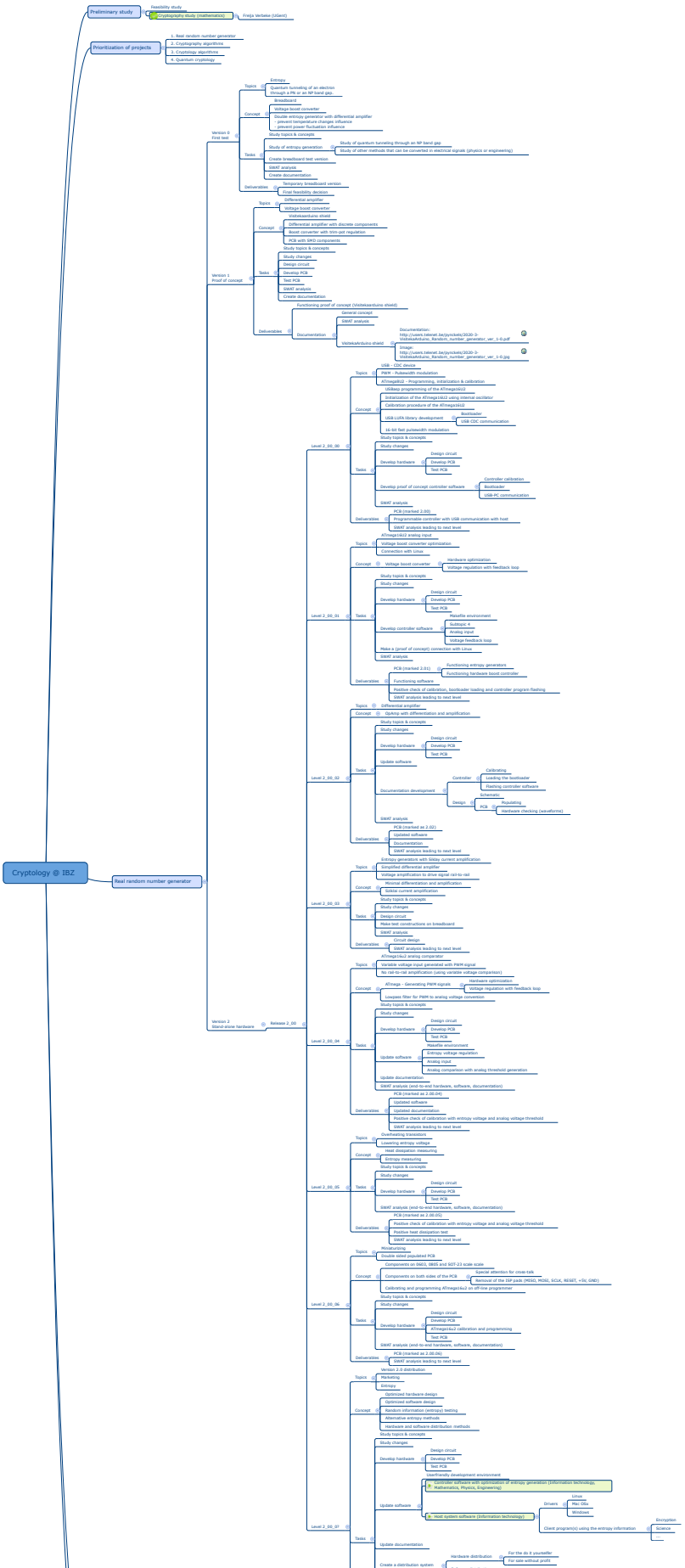
7.2.3. Verification methods

To be written.

8. Conclusion

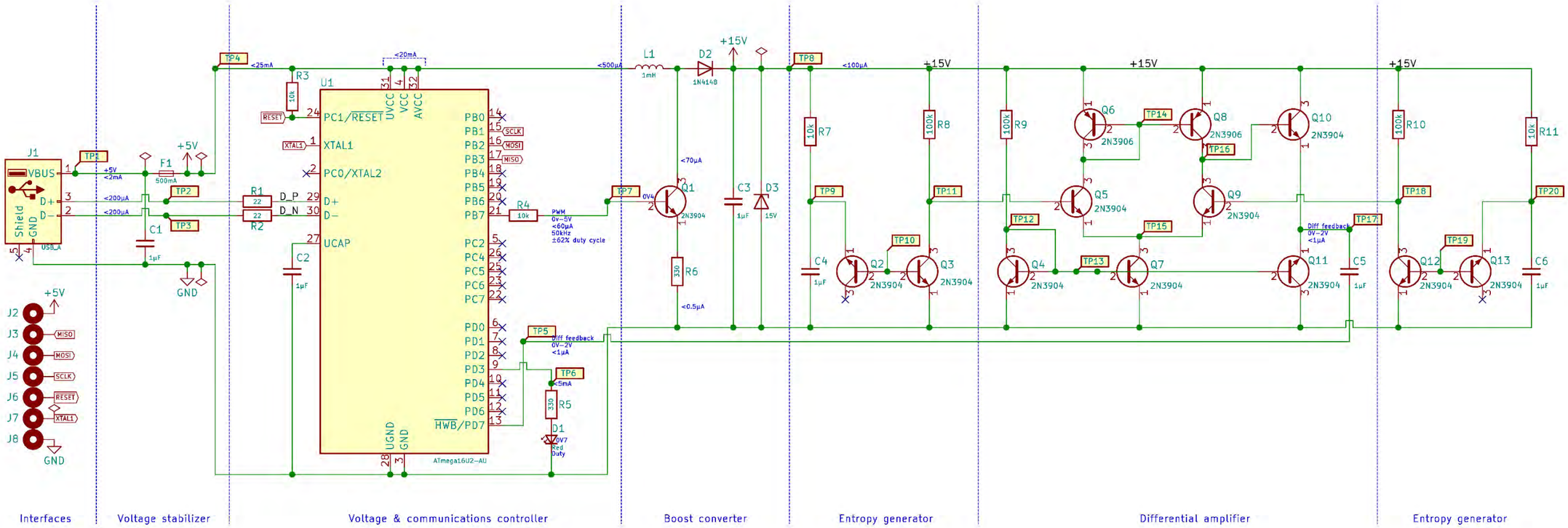
A conclusion will only be possible when the entire project is finished. For the moment, this document and the project are both a work in progress. Tests of the actual hardware are very promising. The software developed thus far is doing a good job but must be optimized. A further mathematical study off the generated random data must be done when the time is ripe. Documentation must be written, but this can only be done when the final versions of the hardware and the software are available.

x 1 - Project outline

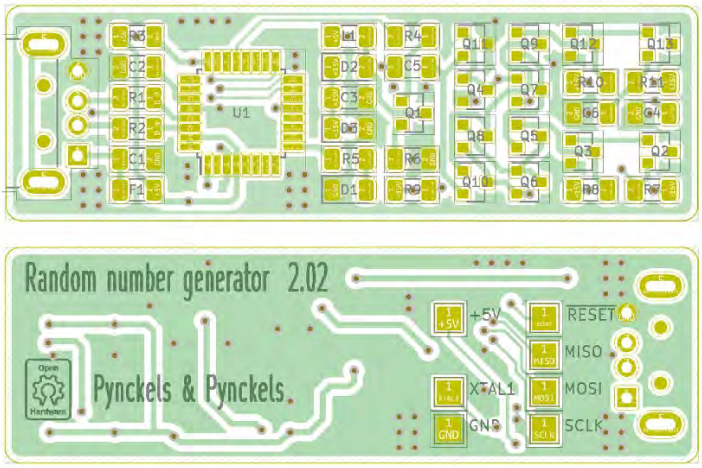


Appendix 2 - Schematic

Schematics of the USB device. Following the state of the project, up-to-date information will be included in this appendix. All test points are shown on the below figure so to permit the reader to measure the signals at the correct place. The name of the components is also mentioned on the PCB image in the bill of materials that can be found in *Appendix 3* .

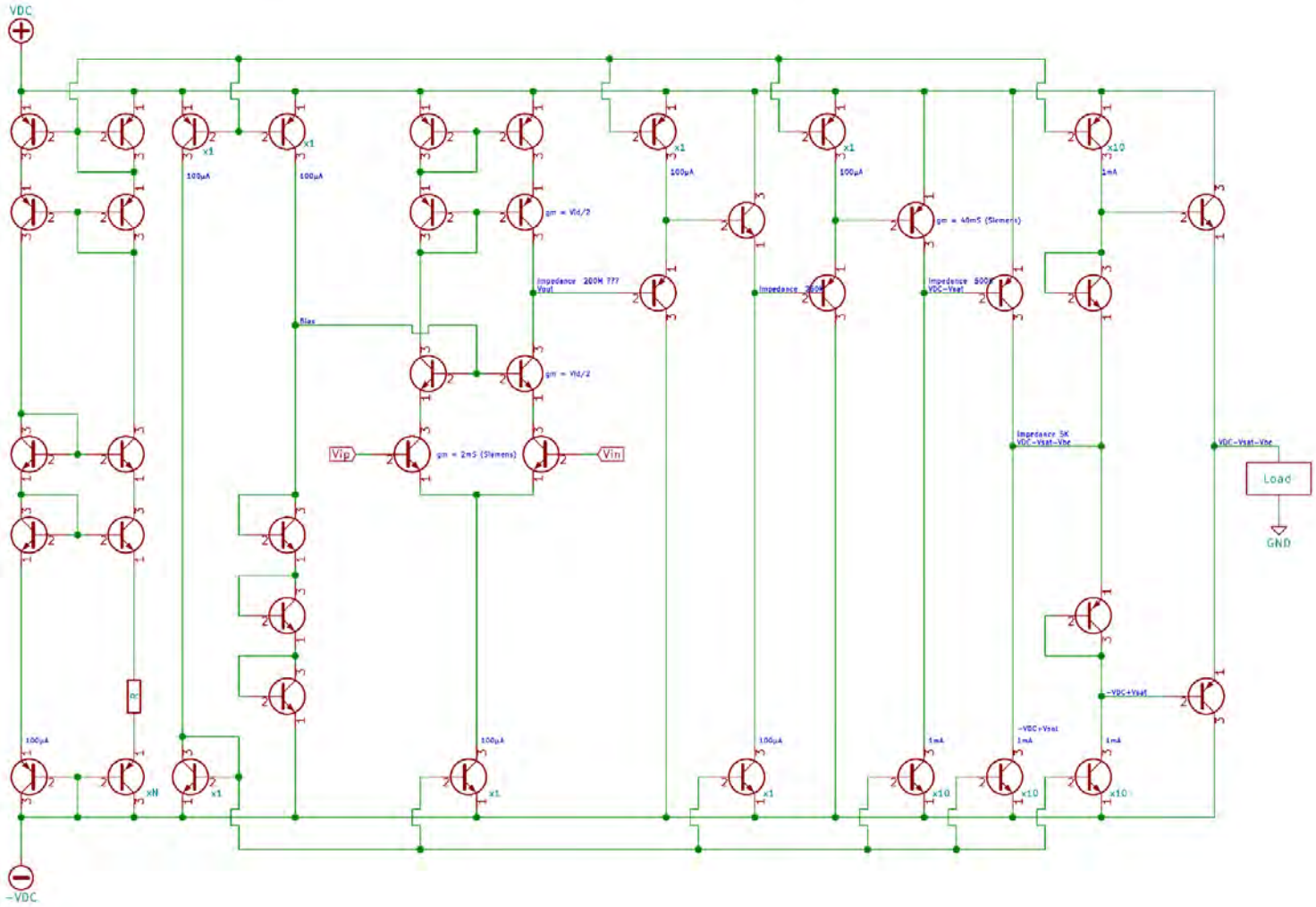


Appendix 3 - Bill of materials



C1	1µF
C2	1µF
C3	1µF
C4	1µF
C5	1µF
C6	1µF
D1	Red
D2	1N4148
D3	15V
F1	500mA
J1	USB_A
L1	1mH
Q1	2N3904
Q10	2N3904
Q11	2N3904
Q12	2N3904
Q13	2N3904
Q2	2N3904
Q3	2N3904
Q4	2N3904
Q5	2N3904
Q6	2N3906
Q7	2N3904
Q8	2N3906
Q9	2N3904
R1	22
R2	22
R3	10k
R4	10k
R5	330
R6	330
R7	10k
R8	100k
R9	100k
R10	100k
R11	10k
U1	ATmega16U2-AU

Appendix 4 - Optimized differential amplifier



High input impedance, low output impedance, few amplification stages, amplification factor is over 1.000.000