

EPHEC

HAUTE ÉCOLE | TECH

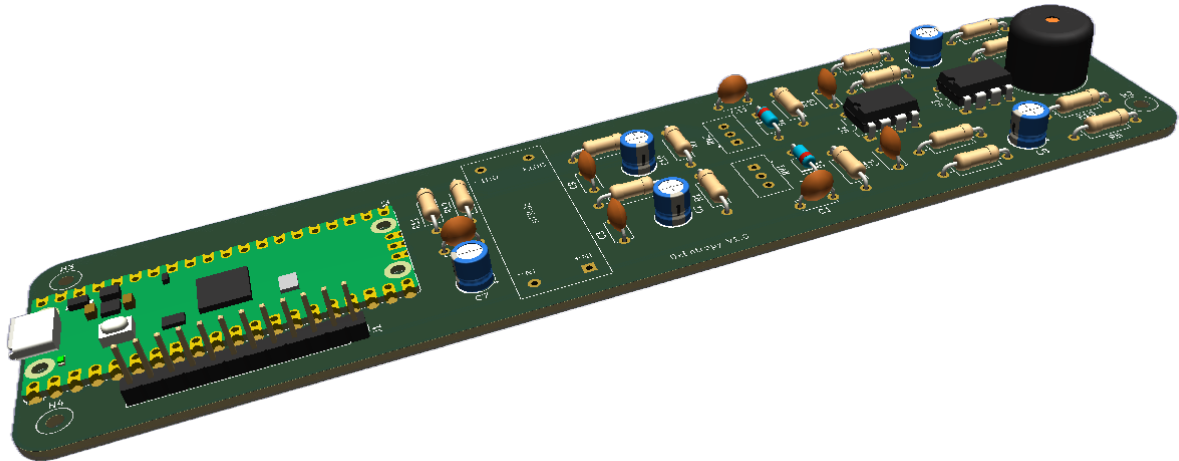
Avenue du Ciseau 15
1348 Louvain-la-Neuve

0xEntropy

Conception et réalisation d'un générateur matériel de nombres aléatoires
portatif basé sur un phénomène physique non-prédictible

Travail de fin d'études présenté en vue de l'obtention du diplôme de bachelier en
Technologies de l'informatique

DE HENAU Dudley



Année académique 2025 - 2026

Rapporteur :
Yves DELVIGNE

Dans cette étude, le rôle de l'IA générative a été :	
<input type="checkbox"/>	A) J'ai écrit l'intégralité de mon texte sans avoir eu recours à un outil d'IA générative ;
<input checked="" type="checkbox"/>	B) J'ai rédigé le contenu de mon travail mais j'ai sollicité un outil d'IA générative pour améliorer
<input checked="" type="checkbox"/>	L'orthographe
<input checked="" type="checkbox"/>	La grammaire
<input checked="" type="checkbox"/>	La syntaxe
<input type="checkbox"/>	C) J'ai consulté un outil d'IA générative pour m'inspirer et puiser des idées de rédaction au niveau du contenu ou de la structure ;
<input type="checkbox"/>	D) J'ai construit des idées que j'ai ensuite soumises à un outil d'IA générative qui m'a aidé à développer mon texte sur base de ces idées ;
<input type="checkbox"/>	E) J'ai sollicité un outil d'IA générative à des fins de traduction ;
<input type="checkbox"/>	F) J'ai confronté plusieurs propositions de contenu produit par l'IA générative pour en sélectionner les passages les plus pertinents, et j'ai édité et amélioré pour la plupart ;
<input type="checkbox"/>	G) J'ai édité et amélioré une proposition de contenu produit par l'IA générative ;
<input type="checkbox"/>	H) Une ou des parties de mon travail ont été intégralement produites au moyen d'un outil d'IA générative sans apport original de ma part.

Remerciements

Je tiens tout d'abord à remercier mon client, Julien Castiaux, de m'avoir proposé ce sujet qui s'est révélé être vraiment passionnant, ainsi que pour son enthousiasme lors de chaque entretien. Je le remercie également d'avoir été super présent au moment du lancement du projet, et pour l'aide précieuse qu'il m'a apportée lors de la rédaction de ce rapport.

Un grand merci également à mon rapporteur, Monsieur Delvigne. Il a toujours été très facile à joindre, s'est rendu disponible quand j'en avais besoin et ses conseils m'ont vraiment bien guidé tout au long de ce projet.

Enfin, je voulais remercier tout particulièrement Patrycja Drewnowska. Merci pour ton soutien moral et pour ton super coup de main sur certaines parties graphiques de mon projet !

Table des matières

Remerciements2

1	Introduction	5
1.1	À quoi ça sert ?	5
1.2	Qu'existe-t-il déjà ?	5
1.3	Ma contribution à cette technologie	6
2	Problème et contexte	7
2.1	Importance de l'aléatoire et limites des générateurs pseudo-aléatoires	7
2.2	Objectif du TFE	7
2.3	Principe général du prototype	7
2.4	Formulation de la problématique	7
2.5	Enjeux techniques	7
2.6	Chaîne complète de génération	8
3	Identification des intervenants	9
3.1	Client	9
3.2	Utilisateur	9
3.3	Synthèse des enjeux	10
4	Identification des fonctionnalités	11
4.1	User Stories, critères d'acceptation, priorités	11
4.2	Matériel	11
4.2.1	Produire de l'aléa matériel réel	11
4.2.2	Être simple à connecter et à utiliser	11
4.2.3	Fonctionner uniquement via USB	12
4.2.4	Être compact et transportable	12
4.3	Logiciel	12
4.3.1	Lecture des données par le microcontrôleur	12
4.3.2	Réduction des biais du flux généré	13
4.3.3	Visualisation et récupération des données produites	13
4.3.4	Évaluation statistique de la qualité de l'aléa	13
4.3.5	Documentation technique du projet	14
4.4	Synthèse des fonctionnalités	14
4.5	Choix techniques et outils envisagés	15
5	Méthodologie	16
5.1	Milestones utilisées	16
5.1.1	Recherche et spécifications	16
5.1.2	Prototype électronique	17
5.1.3	Firmware	17
5.1.4	Validation et tests	17
5.1.5	Version finale	18
5.1.6	Fonctionnement des issues et adaptation du planning	18
5.2	Organisation client	18
5.2.1	Rendez-vous de découverte	18
5.2.2	Rendez-vous de choix technologiques	19
5.2.3	Rendez-vous de présentation du MVP	19
5.2.4	Rendez-vous de présentation du projet final	19
5.2.5	Suivi de l'avancement par messages	19
6	Positionnement de la solution	20
6.1	Solutions existantes	20
6.2	RAVA	20
6.3	Générateur de Rob Seward	20
6.4	Comparaison des solutions	21
6.5	Justification d'une nouvelle solution	21

6.6	Avantages et inconvénients de ma solution	22
6.7	Conclusion du positionnement	22
7	Analyse technique	23
7.1	Objectif de l'analyse technique	23
7.2	Critères de comparaison	23
8	Comparaison des technologies de génération d'aléa	24
8.1	Choix retenu : bruit d'avalanche d'une diode polarisée en inverse	25
8.2	Contraintes du choix retenu	25
8.3	Validation prévue	26
9	Conception de la solution	27
9.1	Matériel	27
9.1.1	Élévation et conditionnement de la tension	28
9.1.2	Génération du bruit et extraction de la composante alternative	28
9.1.3	Amplification du signal analogique	29
9.1.4	Numérisation par comparateur	30
9.1.5	Création d'une masse virtuelle (VREF)	31
9.1.6	Interfaces et retours utilisateurs (Écran et Son)	31
9.1.7	Conclusion de la conception matérielle	32
9.2	Logiciel	32
9.2.1	Échantillonnage matériel indépendant (PIO)	32
9.2.2	Séparation de la charge (Multicœur)	33
9.2.3	Tests de santé en continu et Interface Utilisateur (UI)	34
10	Cas pratique et résolution des défis techniques	35
10.1	Évolution matérielle : De la breadboard au circuit imprimé	35
10.1.1	Les premiers montages et l'identification du bruit	35
10.1.2	Les limites de la breadboard et l'amplificateur inverseur	36
10.1.3	Le passage au circuit soudé	37
10.1.4	L'aboutissement : Le PCB sur mesure	38
10.2	Défis logiciels	38
10.2.1	Le mystère de l'octet 13	38
10.3	Interface graphique : Esthétisme et contraintes de mémoire	39
11	Stratégie de validation et résultats statistiques	40
11.1	Validation matérielle : Contrôle de l'origine du bruit	40
11.2	Validation logicielle : L'outil ent et l'importance du volume de données	40
11.3	Tests avancés : La suite Dieharder (Tests 1 à 17)	40
11.4	Pistes d'amélioration et perspectives futures	41
12	Contraintes législatives et normatives	42
12.0.1	Directive RoHS (Restriction of Hazardous Substances)	42
12.0.2	Directive DEEE / WEEE (Gestion des déchets électroniques)	42
13	Sécurité et intégrité du système	43
13.0.1	Protection de la source d'entropie (Attaques physiques)	43
13.0.2	Maintenance et pérennité à long terme de la solution	43
14	Conclusion	45
14.0.1	Bilan : du concept physique au besoin du client	45
14.0.2	Ce que j'en retiens : réalisation et méthode	45
14.0.3	Comment faire vivre ce projet demain ?	45
14.0.4	Et pour la suite ?	45

*

1 Introduction

1.1 À quoi ça sert ?

Pour toute personne s'intéressant au chiffrement et à la sécurité informatique, la problématique de la génération de nombres réellement aléatoires occupe une place importante. Dans la majorité des systèmes numériques, l'aléa est indispensable pour chiffrer les données stockées et garantir la confidentialité des protocoles de communication, à l'instar du HTTPS.

Lorsqu'un service sécurisé est utilisé sur Internet comme, par exemple, lors d'une connexion à un compte en ligne, l'objectif est d'empêcher une personne non autorisée d'intercepter les données ou d'accéder aux informations personnelles de l'utilisateur. Pour cela, les systèmes de sécurité s'appuient notamment sur des secrets numériques, comme des mots de passe, des jetons d'authentification ou des clés cryptographiques.

Ces éléments doivent être difficiles à deviner et impossibles à prédire. C'est pourquoi la qualité de l'aléatoire utilisé pour les générer est essentielle. Un mot de passe ou une clé de chiffrement générés à partir d'une source prévisible peut fragiliser l'ensemble du système, même si l'algorithme de chiffrement utilisé est solide.

Cependant, un ordinateur est par nature une machine déterministe. Cela signifie qu'à partir d'un même état initial et des mêmes instructions, il produira toujours le même résultat. Ce comportement est ce que l'on souhaite de manière générale : il permet à un programme d'être fiable, reproductible et prévisible dans son fonctionnement. Mais il devient problématique lorsqu'on souhaite produire quelque chose d'imprévisible, comme une suite de nombres réellement aléatoires.

La plupart des ordinateurs utilisent donc des générateurs de nombres pseudo-aléatoires. Ces générateurs produisent des suites de nombres qui semblent aléatoires, mais qui sont en réalité calculées à partir d'une valeur de départ appelée graine. Si cette graine et/ou le fonctionnement du générateur sont connus, il peut devenir possible de retrouver ou de prédire la suite produite.

1.2 Qu'existe-t-il déjà ?

Des solutions existent pour produire ces graines à partir de phénomènes physiques imprévisibles. L'un des exemples les plus connus, et celui qui m'a fait découvrir ce sujet il y a quelques années, est le mur de lampes à lave de Cloudflare. [1].



Figure 1: LavaRAND Cloudflare

À titre d'exemple, l'entreprise Cloudflare utilise un mur de lampes à lave dont les mouvements des bulles sont totalement imprévisibles. Les images de ces lampes, capturées en permanence par des caméras, permettent d'alimenter une source d'entropie utilisée dans leurs systèmes de sécurité.

1.3 Ma contribution à cette technologie

Mon projet a pour but de créer une source d'aléa moins encombrante et ne nécessitant aucune alimentation externe (en dehors de l'USB permettant la communication avec le PC), mais suffisamment fiable pour être exploitée pour du chiffrement. Tout en incluant des tests pour s'assurer de la qualité de l'aléatoire au fil de l'utilisation.

Dans ce rapport, je commencerai par présenter le contexte général du projet ainsi que le problème lié à la génération de nombres réellement aléatoires dans un système informatique. Les différents intervenants et les fonctionnalités attendues du générateur seront ensuite définis, avant d'expliquer l'organisation du travail mise en place durant le projet. Le rapport abordera ensuite le positionnement de la solution, afin de situer le choix d'un générateur matériel basé sur le bruit d'une diode Zener en avalanche par rapport à d'autres méthodes existantes. Une analyse technique permettra ensuite d'étudier les principes électroniques nécessaires, avant de détailler la conception de la solution retenue. Le cas pratique présentera la réalisation concrète du montage et son intégration avec le Raspberry Pi Pico. Enfin, une phase de validation permettra d'analyser les mesures obtenues et d'évaluer le fonctionnement du générateur, avant de conclure sur les résultats, les limites et les améliorations possibles du projet.

2 Problème et contexte

2.1 Importance de l'aléatoire et limites des générateurs pseudo-aléatoires

Dans mon introduction, j'ai présenté l'importance de l'aléatoire dans les systèmes informatiques, en particulier dans le domaine de la sécurité. J'ai également expliqué la limite principale des générateurs pseudo-aléatoires : ils produisent des suites calculées par un algorithme et dépendent d'une valeur initiale appelée graine.

2.2 Objectif du TFE

Mon TFE s'inscrit dans cette problématique. Mon objectif est de concevoir une source d'aléa matérielle, c'est-à-dire basée sur un phénomène physique réel, afin de produire un signal imprévisible pouvant être exploité par un système numérique. Ces sources réellement aléatoires sont appelées TRNG (True Random Number Generator)

2.3 Principe général du prototype

Le projet consiste donc à réaliser un prototype de générateur matériel de nombres aléatoires, ou TRNG, basé sur le bruit électronique produit par une diode Zener utilisée en régime d'avalanche. Ce bruit analogique doit ensuite être conditionné, amplifié et converti en données numériques afin d'être lu par un microcontrôleur, ici un Raspberry Pi Pico.

2.4 Formulation de la problématique

La problématique principale de mon TFE peut donc être formulée de la manière suivante :

Comment concevoir une source matérielle d'aléa, compacte et alimentée par USB, capable de produire un flux de données exploitable par un outil informatique ?

2.5 Enjeux techniques

Cette problématique implique plusieurs enjeux techniques.

Premièrement, le phénomène physique utilisé produit un signal très faible. Le bruit issu d'une diode en avalanche ne peut pas être directement lu par un microcontrôleur. Je dois donc d'abord l'isoler de sa composante continue, puis l'amplifier.

Deuxièmement, le signal obtenu est analogique, alors que le résultat attendu doit être numérique. Je dois donc transformer ce bruit en une suite de bits pouvant être traitée par un programme.

Troisièmement, le flux numérique brut peut contenir des biais ou des déséquilibres statistiques. Le fait que le signal provienne d'un phénomène physique ne garantit pas directement une séquence parfaite. Je dois donc prévoir une étape de traitement logiciel (post-traitement) afin de réduire ces biais et de garantir la production de nombres équitablement distribués.

Quatrièmement, je souhaite réaliser une solution compacte, alimentée uniquement par l'USB. Cette contrainte m'impose de générer localement les tensions nécessaires au fonctionnement de la diode et des circuits de conditionnement, tout en conservant une architecture simple et reproductible.

Enfin, comme le générateur est destiné à produire de l'aléa potentiellement utilisable dans un contexte de sécurité, il ne suffit pas de générer et de traiter un signal variable. Je dois également prévoir une manière d'observer, de tester et d'évaluer la qualité du flux produit.

2.6 Chaîne complète de génération

Le problème de mon TFE ne se limite donc pas à produire du bruit électronique. Il consiste à concevoir une chaîne complète permettant de passer :

```
graph TD; A[d'un phénomène physique imprévisible] --> B[à un signal analogique faible]; B --> C[à un signal conditionné et amplifié]; C --> D[à un flux numérique exploitable]; D --> E[à un flux numérique uniformément distribué]; E --> F[à une source d'aléa utile et testée];
```

d'un phénomène physique imprévisible
↓
à un signal analogique faible
↓
à un signal conditionné et amplifié
↓
à un flux numérique exploitable
↓
à un flux numérique uniformément distribué
↓
à une source d'aléa utile et testée

Cette mise en perspective montre que mon projet repose sur plusieurs domaines complémentaires : l'électronique, le développement et la sécurité informatique.

3 Identification des intervenants

3.1 Client

Le client de mon projet est **Julien Castiaux**, développeur Odoo.

Il est à l'origine de la proposition du sujet. Julien Castiaux souhaitait initialement réaliser lui-même une source matérielle d'aléa, mais il n'a pas eu le temps de développer ce projet. Il a donc proposé cette idée comme sujet de Travail de Fin d'Études.

Son besoin principal est de disposer d'un générateur matériel capable de produire une source d'aléa plus fiable qu'un simple générateur pseudo-aléatoire logiciel. En tant que développeur, son intérêt se situe principalement du côté de l'exploitation des données générées : le système doit pouvoir fournir un flux utilisable par un programme ou par un environnement logiciel.

Dans le cadre de mon projet, son rôle est donc de :

- définir le besoin général ;
- fournir le contexte logiciel et applicatif ;
- exprimer les attentes liées à l'exploitation de l'aléa ;
- servir de référence pour valider l'intérêt fonctionnel du prototype.

Ses objectifs sont :

- obtenir une source d'aléa basée sur un phénomène physique réel ;
- disposer d'un prototype compact et alimenté par USB ;
- pouvoir récupérer une sortie numérique exploitable ;
- obtenir une base technique documentée pouvant être reprise ou améliorée (vision open source).

L'objectif du client est de bénéficier d'une source d'aléa supplémentaire, destinée à être mélangée avec celle fournie nativement par le système hôte. L'enjeu principal est de mettre en place un véritable garde-fou : il s'agit de s'assurer que le système conserve une source d'entropie fiable même en cas de défaillance matérielle ou de compromission du générateur intégré. Mon projet doit donc fournir cette base matérielle externe, tout en restant simple à utiliser depuis l'environnement numérique du client.

3.2 Utilisateur

L'utilisateur visé est la personne ou le système qui exploite le flux d'aléa produit par le générateur.

Dans le cadre de mon TFE, l'utilisateur peut être :

- le client lui-même ;
- un développeur souhaitant intégrer une source d'aléa matérielle dans un programme ;

L'utilisateur n'a pas pour rôle d'intervenir directement sur le phénomène physique ou sur les réglages électroniques internes. Son besoin principal est de recevoir une donnée exploitable, sous une forme numérique compréhensible et intégrable.

Ses objectifs sont :

- obtenir un flux de bits issu d'une source physique ;
- utiliser ce flux dans un programme ;
- pouvoir vérifier que le générateur fonctionne ;
- disposer d'un système simple à connecter, idéalement via USB ;
- avoir accès à des tests ou indicateurs permettant d'évaluer la qualité de l'aléa généré.

L'enjeu principal pour l'utilisateur est donc la facilité d'exploitation. Le système doit masquer la complexité du phénomène analogique et fournir une sortie utilisable dans un contexte logiciel.

3.3 Synthèse des enjeux

Les attentes du client et de l'utilisateur se rejoignent autour d'un objectif commun : obtenir une source d'aléa matérielle exploitable par un système informatique.

Cependant, leurs enjeux ne sont pas exactement les mêmes.

Intervenant	Besoin principal	Enjeu
Client	Disposer d'une source d'aléa matérielle documentée	Avoir une base technique fiable, compacte et réutilisable
Utilisateur	Exploiter simplement le flux produit	Recevoir des données numériques utilisables sans devoir gérer la partie analogique

Table 1: Synthèse des besoins des intervenants

Cette analyse montre que mon projet doit être pensé comme une chaîne complète, allant du phénomène physique jusqu'à l'exploitation logicielle. La valeur du TFE ne réside donc pas uniquement dans la génération du bruit, mais dans ma capacité à rendre ce bruit exploitable, testable.

4 Identification des fonctionnalités

4.1 User Stories, critères d'acceptation, priorités

Les fonctionnalités du projet sont classées en deux catégories principales :

- les fonctionnalités **matérielles**, liées à la génération physique du bruit, à son conditionnement et à l'alimentation du prototype ;
- les fonctionnalités **logicielles**, liées à la lecture du signal, au traitement des données, à l'exploitation du flux et à la validation de l'aléa produit.

Cette séparation permet de distinguer les contraintes liées à l'électronique de celles liées à l'utilisation du générateur par un programme ou un ordinateur.

4.2 Matériel

4.2.1 Produire de l'aléa matériel réel

User Story :

En tant que client, je souhaite disposer d'un générateur matériel de nombres aléatoires, afin d'obtenir une source d'aléa basée sur un phénomène physique réel et de ne pas dépendre uniquement d'un générateur pseudo-aléatoire logiciel.

Critères d'acceptation :

- Le système doit produire de l'aléa à partir d'un phénomène physique non prédictible.
- Le générateur doit être un TRNG, c'est-à-dire un générateur matériel de nombres réellement aléatoires.
- La source d'aléa ne doit pas être uniquement calculée par un algorithme logiciel.
- Le principe physique utilisé doit pouvoir être expliqué et justifié dans le rapport.

Priorité : Très élevée

Complexité : Complexe

4.2.2 Être simple à connecter et à utiliser

User Story :

En tant qu'utilisateur, je souhaite pouvoir utiliser le générateur sans installation complexe, afin de récupérer facilement un flux de données aléatoires.

Critères d'acceptation :

- Le générateur doit pouvoir être connecté simplement à un ordinateur.
- L'utilisateur ne doit pas devoir intervenir directement sur le circuit électronique pour récupérer les données.
- Le système doit pouvoir être utilisé avec une procédure de démarrage simple.
- Le flux de données doit pouvoir être récupéré via une interface standard.

Priorité : Très élevée

Complexité : Moyenne

4.2.3 Fonctionner uniquement via USB

User Story :

En tant que client, je souhaite que le prototype soit alimenté uniquement par USB, afin de disposer d'un système simple à brancher et ne nécessitant pas d'alimentation externe supplémentaire.

Critères d'acceptation :

- Le prototype doit pouvoir être alimenté par le port USB.
- Aucune alimentation externe supplémentaire ne doit être nécessaire pour une utilisation normale.
- Les tensions nécessaires au fonctionnement du circuit doivent être générées localement.
- Le fonctionnement du système doit rester stable lorsqu'il est alimenté par USB.

Priorité : Élevée

Complexité : Complexe

4.2.4 Être compact et transportable

User Story :

En tant que client, je souhaite disposer d'un prototype compact, afin qu'il puisse être facilement transporté, branché et éventuellement intégré dans un environnement informatique.

Critères d'acceptation :

- Le prototype doit rester de dimensions raisonnables.
- Les différents blocs nécessaires au fonctionnement doivent être regroupés dans une architecture simple.
- Le système doit pouvoir être déplacé facilement.
- La solution doit rester reproductible dans le cadre du projet.

Priorité : Élevée

Complexité : Moyenne

4.3 Logiciel

4.3.1 Lecture des données par le microcontrôleur

User Story :

En tant qu'utilisateur, je souhaite que le microcontrôleur lise les données issues du circuit électronique, afin de récupérer le flux généré par la source physique d'aléa.

Critères d'acceptation :

- Le microcontrôleur doit pouvoir lire le signal fourni par le circuit.
- Le programme doit récupérer des données de manière continue.
- Les données lues doivent être exploitables par le reste du traitement logiciel.
- Le programme doit permettre d'observer que le générateur produit bien un flux.

Priorité : Très élevée

Complexité : Moyenne

4.3.2 Réduction des biais du flux généré

User Story :

En tant qu'utilisateur, je souhaite que le flux brut issu du générateur soit traité afin de réduire les biais statistiques, comme un déséquilibre entre les 0 et les 1, pour obtenir une source d'aléa où chaque bit est équitablement probable.

Critères d'acceptation :

- Le système doit pouvoir récupérer un flux brut.
- Un traitement logiciel doit être appliqué au flux brut.
- Le traitement doit viser à réduire les déséquilibres statistiques.
- Le flux traité doit rester issu de la source physique d'aléa.
- Le traitement utilisé doit être expliqué dans le rapport (documenté).

Priorité : Élevée

Complexité : Complexe

4.3.3 Visualisation et récupération des données produites

User Story :

En tant qu'utilisateur, je souhaite pouvoir récupérer les données produites par le générateur et **visualiser** des informations sur leur qualité, afin de vérifier simplement le bon fonctionnement du système.

Critères d'acceptation :

- Les données produites doivent pouvoir être récupérées sur un ordinateur.
- Le logiciel doit permettre d'afficher ou d'enregistrer les données générées.
- L'utilisateur doit pouvoir visualiser des indicateurs simples liés à la qualité des données.
- Le système doit permettre d'identifier un comportement manifestement anormal.

Priorité : Très élevée

Complexité : Moyenne

4.3.4 Évaluation statistique de la qualité de l'aléa

User Story :

En tant qu'utilisateur, je souhaite pouvoir appliquer des **tests statistiques** au flux généré, afin d'évaluer la qualité de l'aléa produit par le générateur.

Critères d'acceptation :

- Le système doit permettre d'enregistrer un échantillon du flux généré.
- Des tests simples doivent pouvoir être appliqués aux données produites.
- Les tests doivent permettre d'observer un éventuel déséquilibre statistique.
- Les résultats doivent pouvoir être analysés dans le rapport.
- Les tests utilisés doivent être documentés.

Priorité : Élevée

Complexité : Moyenne à complexe

4.3.5 Documentation technique du projet

User Story :

En tant que client, je souhaite disposer d'une documentation technique claire, afin de pouvoir comprendre le fonctionnement du générateur, ses limites et les possibilités d'amélioration.

Critères d'acceptation :

- Le fonctionnement général du générateur doit être expliqué.
- Les différents blocs matériels et logiciels doivent être décrits.
- Les choix techniques principaux doivent être justifiés.
- Les limites du prototype doivent être identifiées.
- Des pistes d'amélioration doivent être proposées.

Priorité : Élevée

Complexité : Moyenne

4.4 Synthèse des fonctionnalités

Catégorie	N°	Fonctionnalité	Objectif principal	Outils & Choix envisagés	Priorité	Complexité
Matériel	1	Produire de l'aléa matériel réel	Générer des nombres à partir d'un phénomène physique non prédictible.	Bruit thermique/avalanche (diode/transistor), prototypage sur PCB (ex: KiCad).	Très élevée	Complexe
Matériel	2	Être simple à connecter	Permettre une connexion simple et une récupération directe des données.	Microcontrôleur avec USB natif (ex: RP2040 / Raspberry Pi Pico).	Très élevée	Moyenne
Matériel	3	Fonctionner via USB	Alimenter le prototype uniquement par USB, sans alimentation externe.	Régulateurs de tension intégrés au circuit de conditionnement.	Élevée	Complexe
Matériel	4	Être compact et transportable	Concevoir un prototype de dimensions raisonnables et reproductible.	Intégration globale sur une carte dédiée (composants CMS ou traversants).	Élevée	Moyenne
Logiciel	1	Lecture des données	Lire le signal électronique et récupérer le flux brut en continu.	Programmation en C/C++ et échantillonnage rapide (ex: interface PIO).	Très élevée	Moyenne
Logiciel	2	Réduction des biais	Appliquer un traitement logiciel au flux brut pour réduire les déséquilibres.	Extracteur d'entropie ou hachage cryptographique (ex: SHA-256).	Élevée	Complexe
Logiciel	3	Visualisation et récupération	Récupérer les données sur PC, afficher et valider le flux.	Script Python (bibliothèques <code>pyserial</code> , <code>matplotlib</code>).	Très élevée	Moyenne
Logiciel	4	Évaluation statistique	Appliquer des tests statistiques pour analyser la qualité de l'aléa.	Suites de tests reconnues (ex: NIST SP 800-22, ent).	Élevée	Moyenne à complexe
Logiciel	5	Documentation technique	Décrire le fonctionnement, les choix, les limites et améliorations.	Rapport détaillé (LaTeX / Markdown) incluant les justifications.	Élevée	Moyenne

Table 2: Synthèse des fonctionnalités et des choix technologiques associés

4.5 Choix techniques et outils envisagés

Afin de répondre aux exigences recensées dans les *User Stories* et synthétisées dans le tableau ci-dessus, plusieurs choix technologiques ont été arrêtés. La conception de ce projet s’articule autour d’une approche ouverte, permettant de tester différentes solutions avant validation finale.

Génération physique de l’aléa : Le choix s’oriente vers l’exploitation du bruit analogique d’un composant électronique (comme une diode ou un transistor). Contrairement à des capteurs environnementaux, ces phénomènes quantiques/thermiques offrent un bruit interne autonome et robuste, adapté à la conception d’un TRNG.

Acquisition matérielle (Microcontrôleur) :

Le système devant être compact et fonctionner via USB, le choix se porte sur un microcontrôleur disposant d’un support USB natif et de capacités d’échantillonnage matériel performantes (par exemple, le Raspberry Pi Pico avec son architecture RP2040 et ses blocs PIO permettant de récupérer la valeur d’une pin passivement).

Traitement logiciel et réduction des biais :

Le signal physique brut présentant toujours un léger biais matériel, un traitement est indispensable. Le développement en C/C++ sur le microcontrôleur permettra d’intégrer un algorithme de conditionnement (comme un extracteur d’entropie ou un hachage type SHA-256) directement à la source.

Acquisition PC et Validation statistique :

La récupération des données s’effectuera via un script Python sur l’ordinateur hôte. Pour prouver la qualité cryptographique du flux, des outils d’évaluation mathématiques standardisés (comme la suite NIST SP 800-22 ou l’outil `ent`) seront utilisés, offrant une validation scientifique rigoureuse des résultats.

Conception du schéma et du circuit imprimé (PCB) :

La modélisation du circuit électronique et le routage de la carte matérielle seront réalisés à l’aide d’un logiciel de CAO (Conception Assistée par Ordinateur) tel que KiCad. Ce choix se justifie par le caractère open source et professionnel de cet outil, qui permet de concevoir un prototype sur mesure et compact, intégrant proprement la partie analogique (circuit générateur de bruit) et la partie numérique (microcontrôleur).

5 Méthodologie

J'organise mon travail en suivant directement les milestones que j'ai prédéfinies dans Forgejo au début de mon projet.¹ Le projet n'est donc pas découpé uniquement de façon chronologique, mais plutôt en grandes catégories de travail correspondant aux différentes parties du générateur.

Cette organisation me permet de travailler catégorie par catégorie, tout en gardant une certaine liberté dans l'ordre des tâches. À l'intérieur d'un même milestone, les tâches ne doivent pas forcément être réalisées dans un ordre strict (sauf exception) : je peux commencer par la tâche la plus accessible, la plus urgente ou celle qui débloque le plus rapidement la suite du projet.

L'objectif principal de cette méthode est de garder une vision claire de l'avancement global du projet, tout en restant flexible face aux imprévus techniques.

5.1 Milestones utilisées

Je découpe le projet en cinq milestones ² principales dans Forgejo :

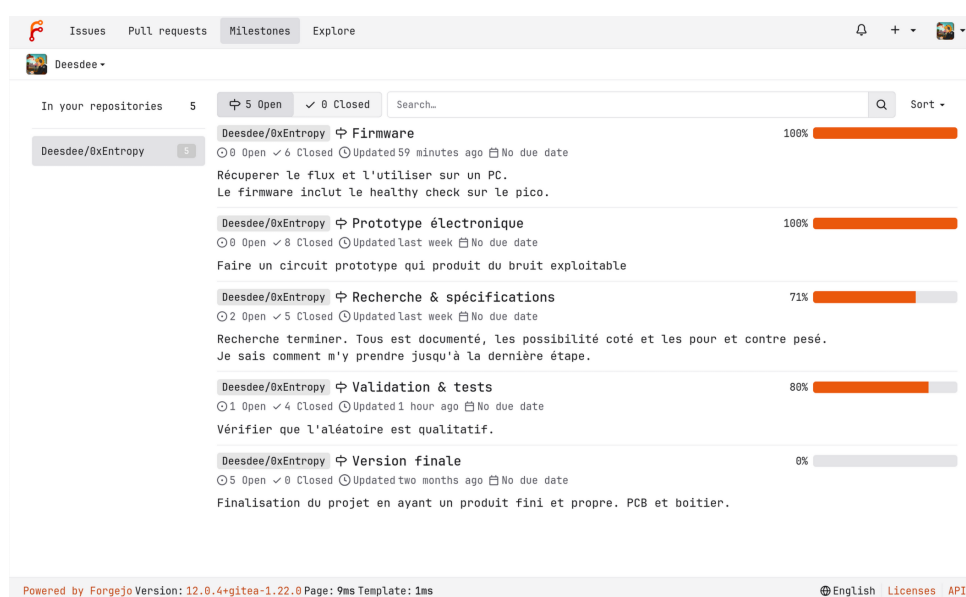


Figure 2: ForgeJo Milestone

Chaque milestone correspond à une grande catégorie de travail. À l'intérieur de chaque milestone, les tâches sont détaillées sous forme d'issues.

5.1.1 Recherche et spécifications

Le milestone **Recherche et spécifications** regroupe les tâches liées à la compréhension du sujet, à la définition de l'architecture générale du générateur et aux choix théoriques importants.

Les principales tâches prévues dans ce milestone sont :

- définir l'architecture globale du TRNG ;
- définir les health tests nécessaires ;
- étudier une architecture CMS³ pour une future version miniaturisée.

Cette étape me permet de poser les bases du projet avant la réalisation pratique. Elle sert à identifier les fonctions principales du générateur, les contraintes techniques et les éléments à prévoir pour obtenir une source d'aléa exploitable.

¹Forgejo est une alternative auto-hébergée et open source à GitHub.

²Un milestone est un outil de gestion de projet permettant de regrouper plusieurs tickets.

³CMS (Composant Monté en Surface) : composant électronique soudé directement à la surface du circuit imprimé (sans perçage), ce qui permet de réduire considérablement la taille de la carte finale.

5.1.2 Prototype électronique

Le milestone **Prototype électronique** est le point central de mon projet, la fin de ce milestone représente mon **MVP**.

Dans ce projet, je définis le MVP comme une **source de bruit utilisable**, capable de produire un signal exploitable permettant de générer une suite de données. À ce stade, le but n'est pas encore d'avoir un système entièrement finalisé, ni d'intégrer tous les health tests ou toutes les corrections statistiques. L'objectif est d'obtenir une première version fonctionnelle du générateur sur le plan matériel, scellée sur une carte de prototypage.

Les principales tâches de ce milestone sont :

- choisir la diode avalanche ;
- concevoir le circuit de polarisation ;
- concevoir l'amplificateur de bruit ;
- choisir un comparateur rapide ;
- dessiner le schéma électronique ;
- réaliser un premier montage ;
- construire un prototype sur breadboard ;
- vérifier que le bruit est exploitable ;
- construire le prototype sur carte de prototypage, correspondant au **MVP**.

Ce milestone est organisé de manière à ce que, lorsqu'il est terminé, le MVP soit atteint. Autrement dit, la validation du milestone **Prototype électronique** signifie que je dispose d'une source de bruit matérielle utilisable, même si le projet n'est pas encore complet.

5.1.3 Firmware

Le milestone **Firmware** regroupe les tâches liées à la récupération numérique du signal et à son transfert vers l'ordinateur.

Les principales tâches prévues sont :

- lire le signal numérique sur le Raspberry Pi Pico ;
- implémenter une lecture rapide des GPIO ;
- générer un flux de bits brut ;
- bufferiser les données ;
- ajouter une sortie USB ;
- mesurer le débit réel post-correcteur.

Ce milestone doit permettre de transformer le signal issu du prototype électronique en un flux numérique récupérable. L'objectif est d'obtenir des données exploitables sur ordinateur, en limitant les pertes et en mesurant les performances réelles du système.

5.1.4 Validation et tests

Le milestone **Validation et tests** regroupe les tâches permettant d'évaluer la qualité du flux produit et d'améliorer son exploitation.

Les principales tâches prévues sont :

- collecter un jeu de données aléatoires assez massif pour subir des tests statistiques intensifs^[1]. (ex : 5GO) ;
- tester les données avec des sites ou outils dédiés ;
- implémenter des health tests ;
- implémenter un correcteur de Von Neumann ;
- implémenter une extraction par hash.

Cette étape doit permettre de passer d'un simple flux brut à un flux mieux caractérisé. Les tests statistiques et les mécanismes de correction ont pour objectif d'identifier les limites du générateur, de réduire certains biais et de documenter la qualité des données produites. Certains points ici impactent directement la section Firmware.

5.1.5 Version finale

Le milestone **Version finale** regroupe les tâches destinées à rendre le projet plus propre, plus robuste et plus facilement réutilisable.

Les principales tâches prévues sont :

- miniaturiser le circuit ;
- concevoir un PCB ;
- ajouter une protection d'alimentation ;
- concevoir un boîtier ;
- publier le schéma et le firmware.

Cette étape correspond à l'amélioration du prototype après validation du fonctionnement général. L'objectif est de préparer une version plus aboutie, mieux intégrée et plus facilement reproductible.

5.1.6 Fonctionnement des issues et adaptation du planning

Même si les milestones donnent une structure claire au projet, l'organisation reste volontairement flexible. Dans une même catégorie, je peux commencer par la tâche que je juge la plus pertinente selon l'avancement du moment. Il n'y a donc pas forcément un ordre strict entre les issues d'un même milestone.

Lorsqu'une tâche avance sans être totalement terminée, je peux ajouter des commits d'avancement dans l'issue concernée. Cela me permet de garder une trace des essais réalisés, des modifications apportées et des décisions prises, même avant la fermeture définitive de l'issue.

Certaines tâches sont également ajoutées au fur et à mesure du projet. Plusieurs besoins apparaissent pendant les choix de composants, les premiers tests ou la conception du circuit. Lorsque je découvre une contrainte ou une amélioration que je n'avais pas anticipée, j'ajoute une nouvelle issue dans Forgejo afin de l'intégrer proprement au suivi du projet.

Cette méthode me permet de garder une organisation claire, tout en conservant assez de souplesse pour m'adapter aux imprévus techniques.

5.2 Organisation client

En plus de l'organisation technique du projet avec Forgejo, je mets également en place un suivi régulier avec le client. L'objectif est de m'assurer que le projet avance dans la bonne direction et que les choix réalisés correspondent bien aux attentes définies au départ.

La communication avec le client s'organise autour de quatre rendez-vous principaux, placés à des moments clés du projet. Ces rendez-vous permettent de valider les grandes étapes, de présenter l'avancement et de discuter des éventuelles adaptations à apporter.

Les quatre rendez-vous principaux sont :

1. **Rendez-vous de découverte**
2. **Rendez-vous de choix technologiques**
3. **Rendez-vous de présentation du MVP**
4. **Rendez-vous de présentation du projet final**

5.2.1 Rendez-vous de découverte

Le premier rendez-vous a pour objectif de comprendre le besoin du client et de définir le cadre général du projet. Durant cet échange, je peux identifier les attentes principales, les contraintes à respecter et le niveau de résultat attendu.

Ce rendez-vous me permet de clarifier le sujet, notamment sur le fait que le projet doit aboutir à un générateur d'aléa portatif avant tout. Il sert également de base pour définir les premières étapes de travail et organiser le projet dans Forgejo.

5.2.2 Rendez-vous de choix technologiques

Le deuxième rendez-vous est consacré à la présentation et à la validation des choix technologiques. Je peux y expliquer les différentes solutions envisagées pour générer du bruit, récupérer le signal et l'exploiter numériquement.

Ce rendez-vous permet de discuter des choix de composants, de l'architecture générale du TRNG et des limites techniques possibles. Il sert également à valider l'orientation du projet avant de passer à la réalisation pratique du prototype électronique.

5.2.3 Rendez-vous de présentation du MVP

Le troisième rendez-vous est centré sur la présentation du **MVP**. Dans ce projet, le MVP correspond à une source de bruit utilisable, capable de produire un signal exploitable et une suite de données récupérable.

À ce stade, l'objectif n'est pas encore de présenter une version entièrement finalisée, avec tous les tests de santé ou tous les traitements statistiques. Le but est surtout de montrer que la partie essentielle du projet fonctionne : produire une source physique de bruit pouvant être utilisée comme base pour la génération d'aléa.

Ce rendez-vous permet de valider une étape importante du projet avant de poursuivre vers les tests, l'amélioration du firmware et la préparation d'une version plus complète.

5.2.4 Rendez-vous de présentation du projet final

Le dernier rendez-vous a pour objectif de présenter l'état final du projet. J'y présente le fonctionnement global du système, les résultats obtenus, les choix réalisés et les limites restantes.

Ce rendez-vous permet de faire le bilan du travail effectué, de comparer le résultat final avec les objectifs définis au départ et de discuter des améliorations possibles pour une future version.

5.2.5 Suivi de l'avancement par messages

En complément des quatre rendez-vous principaux, je maintiens également une communication régulière avec le client par messages texte. Ces échanges permettent de donner des nouvelles de l'avancement général du projet entre deux rendez-vous.

Ce suivi par messages est utile pour signaler les étapes terminées et prévenir en cas de difficulté technique. Il permet également de garder le client informé sans devoir organiser une réunion complète à chaque évolution du projet.

6 Positionnement de la solution

6.1 Solutions existantes

Avant de définir ma solution, j'ai étudié plusieurs approches permettant de générer de l'aléa à partir d'un phénomène physique réel. L'objectif est de situer mon projet par rapport à des solutions existantes et de justifier l'intérêt de développer un nouveau prototype.

Dans l'introduction, j'ai déjà présenté l'exemple de Cloudflare et de son système basé sur un mur de lampes à lave. Cette solution montre qu'il est possible d'utiliser un phénomène physique imprévisible pour alimenter une source d'entropie utilisée dans un contexte de sécurité [1]. Cependant, cette approche n'est pas adaptée à mon projet, car elle repose sur une infrastructure externe importante. Elle n'est donc pas compacte, ni facilement transportable.

Pour mon TFE, les solutions les plus proches sont celles qui utilisent directement du bruit électronique, et plus précisément le bruit d'avalanche d'une diode polarisée en inverse. Deux projets sont particulièrement intéressants à comparer avec ma solution : **RAVA** et le générateur de **Rob Seward**.

6.2 RAVA

RAVA est un générateur matériel de nombres aléatoires open hardware basé sur le bruit d'avalanche [2]. Il s'agit d'une solution proche de mon projet, car elle repose également sur une source physique de bruit électronique. Le projet est documenté, publié et pensé pour être reproductible.

Cette solution présente plusieurs avantages. Elle propose une architecture plus complète qu'un simple montage d'expérimentation. Elle fournit une base matérielle et logicielle déjà structurée, avec une volonté claire de transparence. Le fait que le projet soit open hardware est également intéressant, car cela permet d'étudier son fonctionnement et de s'en inspirer pour comprendre les choix techniques possibles.

RAVA propose aussi deux cœurs d'entropie. Cet aspect peut être intéressant pour mon projet, notamment au moment du choix du matériel. En effet, certains amplificateurs opérationnels intègrent deux amplificateurs dans une même puce, ce qui pourrait faciliter la réalisation de deux chaînes de génération similaires.

Cependant, RAVA ne correspond pas totalement aux objectifs de mon TFE. Une limite importante est que RAVA ne met pas directement l'accent sur des **health tests** embarqués permettant de surveiller l'état de la source de bruit pendant l'utilisation. Or, dans mon projet, cet aspect est important, car il ne suffit pas de produire un signal variable : il faut aussi pouvoir observer, tester et évaluer la qualité du flux produit.

6.3 Générateur de Rob Seward

Le générateur de Rob Seward est une autre solution concurrente pertinente. Il utilise lui aussi le bruit d'avalanche d'une diode pour produire une source d'aléa [3]. Cette solution est intéressante, car elle présente une approche plus expérimentale et plus simple à comprendre. Elle permet de voir concrètement comment un bruit analogique peut être transformé en une suite de bits.

Le principal avantage de cette solution est sa simplicité. Elle constitue une bonne référence pour comprendre le principe de génération de bruit à partir d'une diode en avalanche. Elle montre également que ce phénomène peut être utilisé de manière pratique pour produire un flux numérique.

Cependant, cette solution présente plusieurs limites par rapport à mon cahier des charges. Tout d'abord, elle utilise une alimentation externe secteur. Cela ne correspond pas à l'objectif de mon projet, qui est de réaliser un prototype compact et alimenté uniquement par USB. Ensuite, comme RAVA, cette solution ne propose pas non plus de mécanisme complet de **health check** intégré.

Un point particulièrement intéressant relevé dans cette documentation concerne la dérive du signal dans le temps. Rob Seward explique que sa version précédente avait un problème : après environ trois mois d'utilisation, le signal de **bruit dérivait** et le ratio entre les 0 et les 1 se déséquilibrait [3]. Cette remarque est importante, car elle montre qu'une source physique de bruit peut évoluer avec le temps et qu'un générateur matériel ne doit pas seulement produire de l'aléa à un instant donné.

Ce constat justifie l'intérêt d'intégrer une logique de surveillance dans ma solution. Mon prototype peut donc évoluer vers un système capable de détecter une dérive du signal, de mesurer l'équilibre entre les bits produits et, à terme, de proposer une calibration.

6.4 Comparaison des solutions

Solution	Avantages	Limites
Cloudflare LavaRand	Utilise un phénomène physique réel dans un contexte de sécurité. Montre l'intérêt d'une source d'entropie basée sur un phénomène imprévisible.	Infrastructure lourde, non portable et non adaptée à un prototype compact alimenté uniquement par USB.
RAVA	Solution open hardware, documentée, reproductible et basée sur le bruit d'avalanche. Présence de deux cœurs d'entropie, ce qui peut être intéressant pour comparer ou combiner deux sources de bruit.	Ne met pas directement l'accent sur des health tests embarqués permettant de surveiller la source de bruit pendant l'utilisation.
Rob Seward RNG	Solution simple, pédagogique et basée sur une diode en avalanche. Permet de comprendre concrètement la transformation d'un bruit analogique en une suite de bits.	Utilise une alimentation externe secteur, ce qui ne correspond pas à mon objectif de portabilité. Ne propose pas de health check intégré. Une version précédente a montré une dérive du signal dans le temps, avec un déséquilibre progressif entre les 0 et les 1.
Ma solution	Prototype compact, alimenté uniquement par USB, centré sur une chaîne complète allant du bruit physique jusqu'au flux numérique exploitable. Intègre dès la conception une réflexion sur les tests, la surveillance du flux et une possible calibration.	Nécessite une phase de validation expérimentale. La fiabilité dépendra de la qualité du circuit analogique, du firmware et du traitement logiciel. Le prototype reste moins mature qu'une solution déjà développée et documentée.

Table 3: Comparaison entre les solutions existantes et ma solution

6.5 Justification d'une nouvelle solution

Le développement d'une nouvelle solution est justifié, car les solutions existantes ne répondent pas entièrement aux objectifs définis dans mon projet.

RAVA est une solution open hardware intéressante, documentée et proche de mon sujet. Elle présente notamment l'avantage de proposer deux cœurs d'entropie, ce qui peut être utile pour comparer deux sources de bruit ou envisager une architecture plus robuste. Cependant, elle ne met pas directement l'accent sur des **health tests** embarqués permettant de surveiller l'état de la source pendant l'utilisation.

Le générateur de Rob Seward est plus simple et plus pédagogique. Il constitue une bonne référence pour comprendre le principe d'un générateur basé sur une diode en avalanche. Cependant, il utilise une alimentation externe secteur, ce qui ne correspond pas à mon objectif de réaliser un prototype compact et alimenté uniquement par USB. De plus, cette solution ne propose pas non plus de mécanisme complet de **health check** intégré.

Ces deux solutions montrent que le bruit d'avalanche est une source d'aléa pertinente, mais elles laissent aussi apparaître plusieurs points d'amélioration. En particulier, elles ne répondent pas entièrement à mon besoin de disposer d'un générateur compact, simple à connecter, alimenté uniquement par USB et capable d'être surveillé pendant son fonctionnement.

6.6 Avantages et inconvénients de ma solution

Le principal avantage de ma solution est qu'elle est pensée dès le départ comme une chaîne complète. Le projet ne se limite pas à produire un signal bruité. Il vise à passer d'un phénomène physique imprévisible à un signal analogique conditionné, puis à un flux numérique exploitable par un outil informatique et à un suivi constant de la qualité de ce flux numérique.

Cette approche correspond directement à la problématique de mon TFE : concevoir une source matérielle d'aléa, compacte et alimentée par USB, capable de produire un flux de données exploitable par un outil informatique.

Ma solution présente aussi l'avantage d'être adaptée au cadre du projet. Elle est suffisamment simple pour être comprise, montée et testée dans le cadre d'un TFE, tout en laissant une marge d'évolution vers une version plus robuste. Elle peut notamment intégrer progressivement :

- des tests statistiques ;
- un correcteur de biais ;
- des **health tests** ;
- une calibration du signal ;
- une future miniaturisation en CMS ;
- une publication du schéma et du firmware.

L'intégration possible de **health tests** est un point important. Elle permettrait de vérifier que le générateur reste dans un état de fonctionnement acceptable, par exemple en surveillant le ratio entre les 0 et les 1 ou en détectant une absence anormale de variations. Cette logique de surveillance répond directement au problème de dérive observé dans la solution de Rob Seward.

Cependant, cette nouvelle solution présente aussi des inconvénients. Comme il s'agit d'un prototype, sa fiabilité doit être validée expérimentalement. Le signal produit par la diode doit être mesuré, amplifié correctement, converti en signal numérique, puis testé. La qualité finale du flux dépend donc à la fois du circuit électronique, du firmware et du traitement logiciel. Il s'agit donc d'une nouvelle solution conçue de A à Z. Contrairement à une reprise directe d'un projet existant, il n'est pas possible de s'appuyer sur une base déjà testée dans l'infrastructure retenue pour ma solution.

De plus, même si le prototype fonctionne, il ne peut pas directement être considéré comme une solution cryptographique complète. Il doit d'abord être testé sur de grands volumes de données et comparé à des outils d'analyse statistique. Les **health tests** et les mécanismes de calibration sont donc des éléments importants pour améliorer la robustesse de la solution.

6.7 Conclusion du positionnement

L'analyse des solutions existantes montre que le développement d'un nouveau prototype est pertinent. Les solutions étudiées prouvent que l'utilisation d'un phénomène physique réel, et en particulier du bruit d'avalanche, est une approche valable pour générer de l'aléa. Cependant, elles ne répondent pas complètement aux contraintes de mon projet.

Ma solution se distingue par son objectif de compacité, son alimentation uniquement par USB, son intégration avec un microcontrôleur et sa volonté d'ajouter une surveillance de la qualité du flux produit. Elle ne cherche donc pas seulement à reproduire une solution existante, mais à proposer un prototype adapté aux besoins définis dans ce TFE, avec une attention particulière portée à l'exploitation, aux tests, à la détection de dérive et aux évolutions possibles.

7 Analyse technique

7.1 Objectif de l'analyse technique

L'objectif de cette analyse est d'identifier et de comparer les technologies capables de répondre à la problématique du projet : concevoir une source matérielle d'aléa compacte, alimentée par USB et exploitable par un outil informatique.

Le choix de la source d'aléa est central, car il influence toute la suite du système : polarisation, amplification, conversion en signal numérique, firmware, transmission USB, tests statistiques et surveillance du flux produit.

Plusieurs familles de sources peuvent être envisagées : bruit thermique, bruit de grenaille, bruit d'avalanche, jitter d'oscillateur, oscillateurs en anneau, métastabilité logique, sources optiques, capteurs environnementaux ou encore mémoire au démarrage.

L'analyse technique consiste donc à comparer ces solutions afin de choisir celle qui offre le meilleur compromis pour le prototype. Le but n'est pas de sélectionner la technologie la plus performante en théorie, mais celle qui répond le mieux aux objectifs du client et aux contraintes du TFE : portabilité, simplicité de mise en œuvre, coût raisonnable, possibilité de mesure et validation expérimentale.

Le choix doit également rester compatible avec l'objectif du MVP. La première version du projet doit permettre d'obtenir rapidement une source physique d'aléa fonctionnelle, observable et testable, sans viser directement une solution industrielle ou certifiée.

7.2 Critères de comparaison

Les critères de comparaison sont définis à partir des besoins du client, de la problématique et des fonctionnalités attendues du prototype.

Critère	Justification
Source physique réelle	Le générateur doit produire de l'aléa à partir d'un phénomène physique non déterministe, et non uniquement à partir d'un algorithme pseudo-aléatoire.
Compatibilité USB	Le prototype doit fonctionner sans alimentation externe. La technologie choisie doit donc pouvoir être alimentée à partir de l'USB, éventuellement avec une élévation locale de tension.
Taille	Le dispositif doit rester transportable et intégrable dans un petit montage électronique.
Signal exploitable	La source doit produire un signal mesurable, amplifiable et convertible en données numériques par le système.
Flux continu	Le générateur doit pouvoir produire des données pendant son fonctionnement, et pas uniquement lors d'un événement ponctuel comme le démarrage.
Reproductibilité	Le montage doit pouvoir être refait avec des composants accessibles et des méthodes de mesure vérifiables.
Sensibilité à l'environnement	Une source trop dépendante de la lumière, des ondes radio, de la température, du champ magnétique ou des mouvements est moins maîtrisable et potentiellement manipulable.
Possibilité de validation	Le flux produit doit pouvoir être observé, enregistré et testé statistiquement afin d'évaluer sa qualité.
Coût et disponibilité	Le prototype doit rester réalisable dans le cadre du TFE avec des composants courants et peu coûteux.

Table 4: Critères de comparaison des sources d'aléa

Ces critères permettent de comparer les différentes technologies de manière cohérente avec les objectifs du projet. Ils servent ensuite à justifier le choix d'une source basée sur le bruit d'avalanche d'une diode polarisée en inverse.

Ce choix devra cependant être validé expérimentalement. En effet, même si le bruit d’avalanche est une source physique pertinente, sa qualité dépendra du composant utilisé, de sa polarisation, de l’amplification, du filtrage, de la conversion numérique et des traitements logiciels appliqués au flux brut.

8 Comparaison des technologies de génération d’aléa

Plusieurs familles de technologies peuvent être utilisées pour produire de l’aléa matériel. Elles sont comparées ici selon les critères définis précédemment : source physique réelle, compatibilité avec une alimentation USB, taille, signal exploitable, production d’un flux continu, reproductibilité, sensibilité à l’environnement, possibilité de validation, coût et disponibilité.

#	Famille	Principe	Implémentation	USB	Signal	Flux	Robustesse	Pertinence
1	Bruit thermique	Agitation thermique	Résistance + ampli-op	Oui	Non	Oui	Moyenne	2
2	Bruit shot	Fluctuation du courant	Diode PN / photodiode	Oui	Non	Oui	Moyenne	2
3	Bruit avalanche	Multiplication aléatoire	Diode Zener / Transistor	Oui	Oui	Oui	Élevée	4.5
4	Jitter oscillateur	Variation temporelle	Oscillateur / MCU	Oui	Oui	Oui	Moyenne	3.5
5	Ring oscillators	Oscillateurs en boucle	CMOS / FPGA / ASIC	Oui	Oui	Oui	Élevée	4
6	Métastabilité	État instable logique	Flip-flop	Oui	Oui	Partiel	Moyenne	2
7	Bruit optique	Détection photons	Photodiode	Oui	Oui	Oui	Faible	2.5
8	Bruit RF	Ondes électromagnétiques	Antenne	Oui	Oui	Oui	Faible	1
9	Capteurs MEMS	Micro-variations	Accéléromètre	Oui	Oui	Oui	Faible	1.5
10	Capteurs env.	Variations lentes	Température, etc.	Oui	Non	Non	Faible	1
11	Bruit magnétique	Champ magnétique	Capteur Hall	Oui	Oui	Oui	Faible	1
12	Radioactivité	Particules	Tube Geiger	Non	Oui	Oui	Élevée	1
13	Startup mémoire	État initial	SRAM	Oui	Oui	Non	Élevée	2
14	ADC bruit	Quantification	ADC MCU	Oui	Partiel	Oui	Moyenne	2.5
15	Électrochimique	Réaction chimique	Électrodes	Partiel	Oui	Oui	Faible	1

Table 5: Comparaison des sources d’aléa selon les critères du projet

L’objectif de cette analyse technique (dont les détails par technologie sont développés en Annexe A de l’annexe) est d’identifier et de comparer les solutions capables de répondre à la problématique du projet : concevoir une source matérielle d’aléa compacte, alimentée par USB, et exploitable par un système informatique pour du mixage d’entropie.

Les colonnes du tableau traduisent les critères d’évaluation. La robustesse (immunité aux perturbations extérieures) et la capacité à produire un flux continu sont des facteurs déterminants. La colonne “Pertinence” synthétise cette analyse dans le contexte très spécifique de ce projet, qui implique la réalisation d’un prototype matériel à base de composants discrets (circuit imprimé personnalisé).

Cette comparaison met en évidence que les sources environnementales, RF, magnétiques ou optiques doivent être écartées. Bien qu’elles produisent des variations, elles sont trop dépendantes de l’environnement extérieur, offrent un débit souvent trop faible, et surtout, sont vulnérables aux attaques par injection d’interférences (manipulation du signal).

Le cas des Ring Oscillators (Oscillateurs en anneau)

Les solutions purement numériques, telles que les oscillateurs en anneau (Ring Oscillators), présentent une robustesse élevée et d’excellentes performances. Elles constituent d’ailleurs le standard de l’industrie pour les générateurs intégrés directement dans le silicium des processeurs. Cependant, leur pertinence baisse dans le cadre d’un prototype “maison”.

Pour garantir une véritable source d’entropie, un Ring Oscillator doit être implémenté au sein d’un composant très intégré (comme un FPGA⁴), où les temps de propagation sont de l’ordre de la picoseconde et isolés des perturbations externes [4]. Tenter de reproduire ce montage avec des portes logiques discrètes sur un circuit imprimé (PCB) expose le système à un couplage parasite déterministe (diaphonie entre les pistes et bruit d’alimentation). Ce phénomène, connu sous le nom de verrouillage par injection (injection locking), force les oscillateurs à se synchroniser, ce qui détruit l’aléa (la gigue de phase) au profit d’un

⁴FPGA : *Field-Programmable Gate Array*, circuit intégré logique programmable permettant de réaliser des fonctions numériques matérielles personnalisées à l’échelle du silicium.

comportement déterministe [5]. De plus, valider mathématiquement l'entropie d'un tel montage discret nécessiterait une instrumentation de laboratoire hors de portée de ce projet.

Le choix du bruit d'avalanche

Le bruit d'avalanche (généré par la jonction PN d'un transistor ou d'une diode Zener polarisée en inverse) apparaît ainsi comme le meilleur compromis technique et pédagogique pour ce travail de fin d'études. Il repose sur un phénomène quantique robuste, générant un bruit blanc à large bande, très peu sensible aux variations environnementales courantes. Contrairement aux solutions purement numériques, il peut être implémenté de manière très efficace avec des composants électroniques standards, peu coûteux et très compacts. Surtout, cette technologie offre un avantage majeur pour le prototypage et la validation : elle produit un signal analogique brut, chaotique et de forte amplitude une fois amplifié. Ce signal est visuellement vérifiable à l'oscilloscope (prouvant l'existence de la source physique) avant même d'être numérisé par un microcontrôleur. Elle répond donc parfaitement à l'objectif de fournir un garde-fou matériel transparent, vérifiable et indépendant.

8.1 Choix retenu : bruit d'avalanche d'une diode polarisée en inverse

La technologie retenue pour le prototype est le bruit d'avalanche produit par une diode polarisée en inverse.

Ce choix s'inscrit dans un cadre plus large de conception des générateurs d'aléa matériels. Le standard américain NIST SP 800-90B, **ainsi que la méthodologie européenne de référence AIS 31 [6], définissent** une source d'entropie comme un système composé d'un phénomène physique, d'un conditionnement du signal et de mécanismes de validation [7]. Cette approche correspond directement à l'architecture envisagée dans ce projet : production d'un bruit physique, amplification, conversion numérique et analyse statistique.

Contrairement à ce que l'on pourrait attendre, la littérature scientifique et les standards **comme le NIST ou l'AIS 31** ne proposent pas de benchmark direct comparant les différentes sources physiques d'aléa (bruit thermique, jitter, oscillateurs, avalanche, etc.). Les évaluations portent généralement sur des générateurs complets, intégrant à la fois la source, le conditionnement et le post-traitement [7].

Dans ce contexte, le choix d'une source ne peut pas reposer uniquement sur des performances théoriques. Il doit prendre en compte des critères pratiques tels que la facilité de mise en œuvre, la robustesse du signal, la compatibilité avec l'architecture globale et la capacité à être validé expérimentalement.

Le bruit d'avalanche présente un compromis intéressant. Il repose sur un phénomène physique réel et exploitable, peut être mis en œuvre avec des composants simples et reste compatible avec une alimentation USB moyennant une élévation locale de tension. De plus, il fournit un signal analogique d'amplitude suffisante pour être amplifié et numérisé sans nécessiter une instrumentation trop complexe.

Cette solution correspond également à l'approche MVP du projet. Elle permet de tester rapidement la chaîne complète de génération d'aléa sans imposer une architecture trop complexe. Le prototype peut ainsi être construit progressivement : source physique, conditionnement analogique, conversion numérique, export USB, puis validation statistique.

Le choix du bruit d'avalanche ne signifie pas que cette solution est optimale dans tous les contextes. Il s'agit du meilleur compromis identifié pour ce projet, en tenant compte des contraintes de temps, de coût, de portabilité et de validation expérimentale.

8.2 Contraintes du choix retenu

Même si le bruit d'avalanche constitue une solution pertinente, plusieurs contraintes techniques et défis de conception doivent être pris en compte.

La première contrainte concerne la variabilité des composants (tolérance matérielle). Le bruit d'avalanche est un phénomène physique sensible : deux diodes de la même référence peuvent présenter des caractéristiques de bruit différentes en amplitude et en fréquence. Une phase de caractérisation expérimentale est donc nécessaire pour s'assurer que le signal obtenu reste exploitable et pour garantir la reproductibilité du montage.

Le deuxième défi technique concerne la tension de polarisation. Le fonctionnement en avalanche nécessite une tension inverse suffisante (généralement plus élevée que les tensions standards). Dans le cadre

d'une alimentation USB stricte (5V), cela impose l'utilisation d'un circuit d'élévation de tension (boost converter). Ce point est critique, car les circuits à découpage génèrent un bruit parasite périodique (déterministe). Le circuit doit donc être conçu et filtré avec un soin extrême afin d'éviter que ce bruit déterministe ne vienne altérer l'entropie du bruit naturel de la diode.

La troisième contrainte porte sur la pureté statistique du flux numérique brut. Il est fréquent de penser qu'une source matérielle (TRNG) produit naturellement un signal parfait. En réalité, bien que le phénomène d'avalanche de la diode soit fondamentalement imprévisible, l'électronique qui l'entoure introduit inévitablement des imperfections.

Par exemple, le Raspberry Pi Pico génère du bruit numérique sur son alimentation, et la moindre asymétrie dans la tension de référence (les 2.5V nécessaires aux étages de filtrage et de comparaison) va légèrement favoriser les 1 ou les 0 au niveau du comparateur. Le flux brut présente donc fatalement des biais d'asymétrie ou de légères corrélations matérielles. Conformément aux recommandations du NIST SP 800-90B [7] et à la littérature sur les générateurs d'aléa matériels [8], le signal ne peut donc pas être considéré comme directement exploitable sans une phase de conditionnement logiciel (post-traitement ou « blanchiment ») visant à lisser ces biais physiques.

8.3 Validation prévue

La validation du choix technique s'appuie sur une démarche expérimentale alignée avec les recommandations du NIST SP 800-90B. Pour lever toute ambiguïté, il convient de distinguer les évaluations lourdes réalisées hors ligne (en environnement de test) de la surveillance allégée effectuée en temps réel pendant l'exploitation.

Dans un premier temps, le signal analogique sera observé afin de vérifier la présence d'un bruit exploitable après amplification. Ensuite, un flux numérique brut sera acquis afin d'analyser sa distribution, sa stabilité et ses éventuelles corrélations.

Une série de tests statistiques complexes sera ensuite appliquée hors ligne sur des captures de données. Parmi ceux-ci, l'outil Dieharder sera utilisé pour évaluer la qualité du flux binaire produit [9]. Dieharder est une suite de tests statistiques appliquant différentes méthodes (tests de fréquence, de séquences, de corrélations, etc.) afin de vérifier si un flux présente un comportement compatible avec celui d'une source aléatoire. Il permet notamment de détecter des biais, des structures répétitives ou des dépendances entre bits qui ne seraient pas visibles à l'œil nu.

Dans ce projet, Dieharder est utilisé comme un outil de validation et de calibrage en laboratoire : il ne permet pas de prouver qu'un générateur est parfaitement sûr, mais il constitue un moyen efficace pour identifier des défauts statistiques évidents et comparer différentes versions du système (signal brut, signal après post-traitement, réglages analogiques de la diode, etc.) [9]. Ces tests seront complétés par des méthodes d'estimation de l'entropie minimale (min-entropy) sur de larges échantillons, recommandées par le NIST SP 800-90B [7].

Cependant, conformément à l'exigence du NIST SP 800-90B d'intégrer des tests de santé en continu (continuous health tests), le système ne se contente pas de validations hors ligne. Puisque les suites comme Dieharder sont trop gourmandes en données pour analyser le flux à la volée, une évaluation légère de l'entropie est réalisée en direct pendant l'exploitation de la puce. Ce test en temps réel permet de vérifier en continu que la source délivre un signal d'une qualité optimale (avec une entropie approchant les 7,9999 bits par octet), garantissant ainsi la détection immédiate d'une éventuelle défaillance matérielle.

9 Conception de la solution

Si le bruit d'avalanche est la "matière première" de ce projet, sa transformation en un flux de données exploitable constitue le véritable défi technique du montage. Passer du monde analogique au monde numérique demande une précision extrême pour ne pas dénaturer l'aléa source. Cette difficulté est un pilier historique de la discipline : dès 1974, des brevets industriels détaillaient déjà les spécificités des circuits d'interface nécessaires à la numérisation du bruit [10]. Ce chapitre détaille comment cette solution est remise au goût du jour avec les moyens techniques actuels.

9.1 Matériel

L'analyse technique a permis d'identifier le bruit d'avalanche comme la source d'entropie la plus adaptée pour ce projet. Cependant, comme expliqué précédemment, la conception d'un TRNG ne se limite pas au choix du composant physique. Il faut concevoir une chaîne complète d'acquisition et de traitement du signal.

Ce chapitre détaille la conception matérielle du prototype « 0xEntropy v1.0 ». L'objectif est de justifier chaque bloc du circuit électronique (3) en démontrant comment il répond aux contraintes établies : alimentation USB (5V), isolation des bruits déterministes, amplification du signal et numérisation sécurisée pour le microcontrôleur.

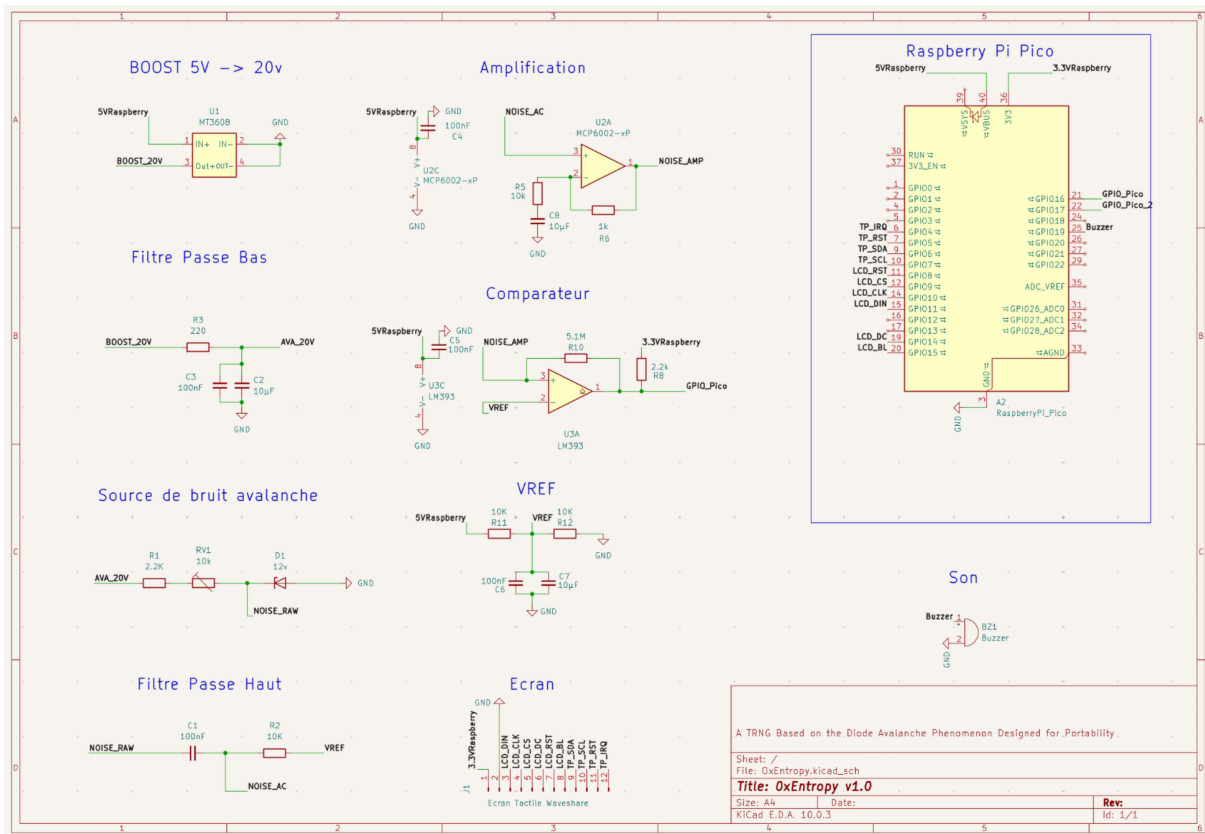


Figure 3: Schéma Complet OxEntropy V1.0

Pour faciliter la compréhension de cette conception, le circuit a été découpé en plusieurs blocs fonctionnels, détaillés ci-dessous. Aussi, seul un cœur d'entropie est décrit pour faciliter la compréhension. Le schéma intégrant le deuxième cœur d'entropie est disponible sur le GIT du projet.

9.1.1 Élévation et conditionnement de la tension

Le Raspberry Pi Pico et la connexion USB fournissent une tension de 5V. Or, le phénomène d'avalanche nécessite une tension inverse plus élevée pour s'amorcer correctement, nécessitant une tension d'alimentation supérieure à la tension de claquage de la diode (12V pour la diode choisie).

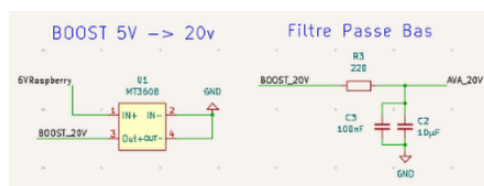


Figure 4: Circuit d'élévation de tension et filtrage

Justification des choix :

Pour résoudre ce problème de tension, j'ai intégré un convertisseur Boost basé sur le composant MT3608 (U1). Ce circuit permet de transformer le 5V du Raspberry Pi en 20V (BOOST_20V), offrant ainsi une marge suffisante pour polariser la diode.

Cependant, les régulateurs à découpage comme le MT3608 génèrent un bruit haute fréquence fortement déterministe lié à leur fréquence de commutation. Si ce bruit parasite atteint la diode, il se mélangera au bruit quantique aléatoire et détruira l'entropie du générateur. C'est pourquoi j'ai conçu un **filtre passe-bas** (composé de la résistance R3 de 220Ω et des condensateurs C2 de 10μF et C3 de 100nF) placé immédiatement après le Boost. Ce filtre lisse la tension et fournit une alimentation de 20V continue et « propre » (AVA_20V) au circuit de génération d'aléa.

9.1.2 Génération du bruit et extraction de la composante alternative

Le cœur d'entropie est constitué du circuit de polarisation de la diode et d'un filtre passe-haut pour extraire le signal utile.

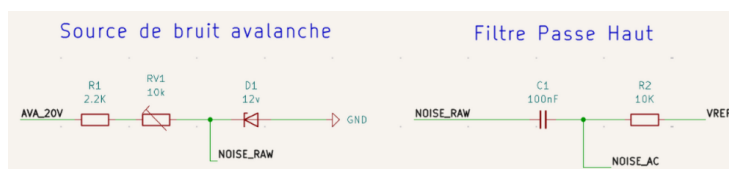


Figure 5: Bloc Source d'avalanche, Filtre Passe-Haut

Justification des choix :

La diode Zener D1 de 12V est polarisée en inverse grâce à la tension de 20V filtrée [11]. J'ai fait le choix d'intégrer un potentiomètre (RV1 de 10kΩ) en série avec la résistance de protection R1 (2.2kΩ). Cela permet d'ajuster finement le courant traversant la diode en faisant chuter la tension. L'avalanche est un phénomène sensible : le potentiomètre permet de calibrer le montage pour trouver le point de fonctionnement exact où l'amplitude du bruit est maximale (cela même si le circuit est calibré pour fonctionner sur de basses amplitudes).

Le signal brut obtenu (NOISE_RAW) est superposé à une tension continue (DC) proche de 12V (la tension de claquage de la Zener). Pour être traité par des amplificateurs alimentés en 5V, ce signal doit être recentré. J'utilise un **filtre passe-haut** (C1 de 100nF et R2 de 10kΩ) qui bloque la composante continue et ne laisse passer que les variations aléatoires (AC).

Le signal alternatif est ensuite superposé à une masse virtuelle (VREF fixée à 2.5V via la résistance R2). Ainsi, le signal de bruit (NOISE_AC) oscille positivement et négativement autour de 2.5V, restant parfaitement au centre de la plage de fonctionnement de l'amplificateur opérationnel de l'étage suivant.

9.1.3 Amplification du signal analogique

Le bruit récupéré à la sortie du filtre est de l'ordre de quelques millivolts oscillant autour de 2.5V. Il est inexploitable en l'état par un outil numérique.

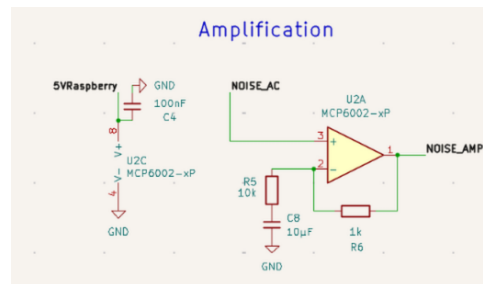


Figure 6: Bloc Amplification

Justification des choix :

Les résistances R6 (1k Ω) et R5 (10k Ω) fixent le gain de l'amplificateur. Dans cette configuration, le gain est défini par la formule $G=1+R5/R6$, ce qui nous donne ici un gain très faible de 1,1.

Ce choix d'un gain bas est volontaire. En électronique, les amplificateurs sont limités par ce qu'on appelle le « produit gain-bande passante » : plus on demande un gain élevé à l'amplificateur, plus il devient lent. En gardant un gain très faible, on s'assure que le composant reste suffisamment rapide pour capturer et restituer les variations très rapides (hautes fréquences) du bruit d'avalanche, sans les déformer.

Le condensateur C8 (10 μ F), placé avec R5, a simplement un rôle de protection : il empêche d'amplifier la tension de repos (2.5V) pour éviter de saturer le signal, garantissant que seul le bruit est traité.

Enfin, la puce MCP6002 a été choisie car elle est de type Rail-to-Rail. Cela lui permet d'exploiter toute sa plage d'alimentation (de 0V à 5V) pour générer son signal de sortie, sans couper les crêtes du bruit.

9.1.4 Numérisation par comparateur

Cette étape consiste à traduire ce signal analogique aléatoire en un flux binaire (0 ou 1) lisible par le Raspberry Pi Pico.

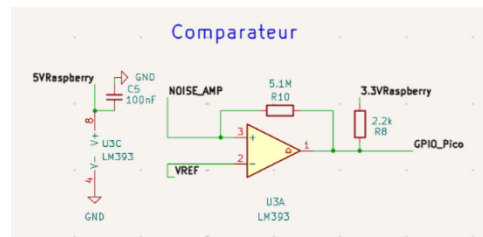


Figure 7: Bloc Comparateur et connexion Pico

Justification des choix :

Pour numériser le signal sans dépendre exclusivement du convertisseur analogique-numérique (ADC) du microcontrôleur (qui est lent et génère son propre bruit de quantification déterministe), j'ai opté pour un comparateur matériel LM393 (U3A).

Le comparateur compare le bruit amplifié (NOISE_AMP, sur l'entrée non-inverseuse) à la tension de référence (VREF de 2.5V, sur l'entrée inverseuse). * Si le bruit dépasse 2.5V, la sortie passe à l'état haut (1). * S'il est inférieur à 2.5V, la sortie passe à l'état bas (0).

Une précaution importante de conception réside dans l'utilisation de la résistance de tirage (pull-up) R8 de 2.2k Ω . Celle-ci est reliée au 3.3V du Raspberry Pi Pico et non au 5V. Cela garantit que le signal sortant ne dépasse jamais 3.3V, protégeant ainsi la broche GPIO du microcontrôleur, qui ne tolère pas le 5V en entrée. Enfin, la résistance R10 de 5.1M Ω introduit une légère hystérésis au comparateur, évitant des oscillations parasites instables autour du seuil de basculement.

9.1.5 Création d'une masse virtuelle (VREF)

Pour rendre le prototype autonome et permettre un suivi en temps réel de la génération d'aléa (état du système, débit, alertes), j'ai besoin d'une interface utilisateur directe. Étant donné que le Raspberry Pi Pico ne dispose nativement d'aucun moyen de retour visuel complexe ou sonore, j'ai intégré un écran tactile et un buzzer dédiés à cette fonction.

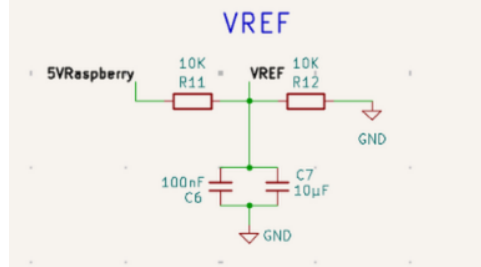


Figure 8: Bloc de tension de référence VREF

Justification des choix :

Pour résoudre ce problème, j'ai créé une « masse virtuelle » ou tension de référence (VREF) fixée à 2.5V. Celle-ci est générée par un pont diviseur de tension classique, composé de deux résistances de valeur identique (R11 et R12 de 10kΩ), divisant le 5V par deux.

Cependant, le 5V provenant d'un port USB ou du Raspberry Pi est généralement très bruité (bruit numérique de l'ordinateur, commutations du microcontrôleur). Si ce bruit contaminait VREF, il s'ajouterait au signal aléatoire à travers tout le reste du montage. Pour garantir une référence parfaitement stable, j'ai ajouté un filtrage capacitif composé d'un condensateur de découplage de 100nF (C6) pour absorber les parasites haute fréquence, et d'un condensateur de 10μF (C7) pour stabiliser la tension face aux variations plus lentes [12].

Cette tension VREF propre et stable est ensuite distribuée au filtre passe-haut (pour recentrer le signal brut à 2.5V) et au comparateur (comme seuil de basculement).

9.1.6 Interfaces et retours utilisateurs (Écran et Son)

Pour rendre le dispositif autonome et permettre un suivi en temps réel de la génération d'aléa (état du système, débit, alertes), j'ai besoin d'une interface utilisateur directe. Étant donné que le Raspberry Pi Pico ne dispose nativement d'aucun moyen de retour visuel complexe ou sonore, j'ai dû intégrer un écran tactile et un buzzer dédiés à cette fonction.

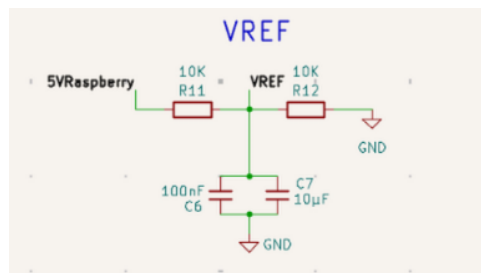


Figure 9: Bloc de tension de référence VREF

Justification des choix :

Afin de rendre le prototype "0xEntropy v1.0" autonome et interactif, des interfaces utilisateurs ont été prévues sur la carte. Le schéma intègre le câblage d'un **écran tactile** (via le bus SPI et des broches dédiées au tactile) permettant de visualiser l'état du système, le débit, ou les résultats des tests d'entropie en temps réel. Un **buzzer** (BZ1) piloté par un GPIO a également été ajouté pour offrir un retour auditif (feedback) lors des interactions ou pour signaler d'éventuelles erreurs matérielles.

9.1.7 Conclusion de la conception matérielle

La conception de ce circuit répond directement aux contraintes physiques de la production d'aléa. Chaque étage a une fonction précise : préserver la nature aléatoire du bruit d'avalanche (élévation de tension et filtrage strict de l'alimentation), assurer l'intégrité du signal (amplification avec blocage de la composante DC) et adapter la tension pour communiquer en toute sécurité avec le microcontrôleur (comparateur tiré à 3.3V).

Cette base matérielle constitue le cœur du prototype. Elle doit fournir au logiciel (firmware) un flux de données brutes exploitable, ce qui permettra ensuite d'entamer les phases de tests statistiques et de post-traitement abordées dans la suite de ce rapport.

9.2 Logiciel

La conception logicielle du générateur a été pensée pour répondre à une contrainte majeure : ne jamais ralentir le flux d'acquisition de l'aléa ni en altérer la qualité statistique, tout en maintenant une interface utilisateur réactive. Pour atteindre cet objectif, le firmware exploite pleinement l'architecture matérielle et multicœur du microcontrôleur RP2040 du Raspberry Pi Pico.

Seules les quatre parties fondamentales du système sont détaillées dans ce chapitre. Bien que mon programme intègre des fonctionnalités et options supplémentaires, j'ai fait le choix de me concentrer ici sur les éléments vitaux. L'intégralité du code source (incluant les drivers et les utilitaires) est disponible sur le dépôt Git du projet [13]. Enfin, par souci de transparence, les portions de code que j'ai entièrement implémentées par moi-même sont regroupées en Annexe C.

9.2.1 Échantillonnage matériel indépendant (PIO)

L'une des particularités du RP2040 est la présence de blocs PIO (*Programmable I/O*). Plutôt que d'utiliser une interruption classique ou une boucle logicielle, l'acquisition du signal numérique issu du comparateur matériel est déportée sur une machine à états indépendante.

```
; Extrait de noise.pio :  
; Boucle infinie qui lit 1 bit sur la broche configurée et l'envoie dans le registre  
loop:  
    in pins, 2  
    jmp loop
```

Justification des choix :

L'utilisation d'une boucle logicielle classique (dans le code C) aurait introduit du *jitter* (gigue temporelle) en raison des cycles d'horloge perdus ou des interruptions du système, ce qui aurait faussé l'échantillonnage temporel du bruit. Le choix d'utiliser le module PIO garantit un échantillonnage d'une régularité absolue et totalement matériel. Les bits lus sont poussés de manière autonome par blocs de 32 dans un registre FIFO, sans jamais monopoliser le processeur central. Ce choix technique assure que le flux binaire capturé reflète fidèlement le phénomène physique sans aucune distorsion temporelle.

9.2.2 Séparation de la charge (Multicœur)

Le traitement des données brutes nécessite l'application d'une fonction de hachage lourde (SHA-256) pour le « blanchiment » (*whitening*) des biais matériels. Parallèlement, le système doit gérer un écran tactile capacitif complexe et afficher des statistiques.

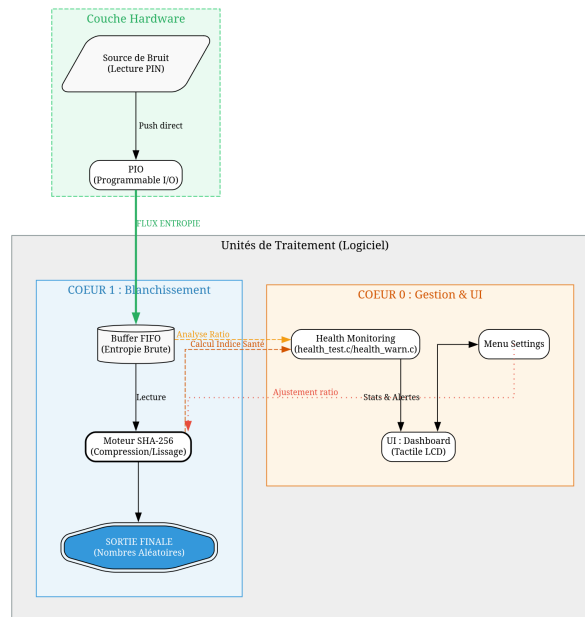


Figure 10: Diagramme de répartition des tâches sur l'architecture multicœur du RP2040

Justification des choix :

Pour gérer ces deux besoins contradictoires (un calcul intensif et régulier d'un côté, une interface graphique lente de l'autre), j'ai opté pour une architecture logicielle scindée de manière asymétrique sur les deux cœurs du RP2040 :

Le Cœur 1 est dédié au traitement critique :

Il fonctionne en "circuit fermé". Il récupère les bits depuis le registre du PIO, calcule les statistiques brutes (Tests de Santé) et applique l'algorithme SHA-256 pour conditionner le signal.

Le Cœur 0 gère la supervision et les E/S :

Il s'occupe de l'affichage SPI de l'écran, de l'interaction tactile (`Screen_tactile.c`), de la communication USB avec l'ordinateur et de la surveillance des alertes.

Cette séparation stricte empêche le rafraîchissement coûteux de l'écran (qui prend plusieurs millisecondes) de provoquer la perte du moindre bit d'entropie généré par le Cœur 1. Le rafraîchissement de l'écran est également limité dans le but de ne pas surcharger le Cœur 0 et, par la même occasion, d'éviter de ralentir le système.

9.2.3 Tests de santé en continu et Interface Utilisateur (UI)

Lors de mes recherches sur l'état de l'art, j'ai constaté que la norme NIST SP 800-90B exige qu'un TRNG matériel puisse détecter une défaillance immédiate de sa source de bruit (par exemple, si ma broche d'acquisition se retrouve court-circuitée à la masse ou si le composant d'avalanche grille). Il me paraissait essentiel d'intégrer cette exigence de sécurité à mon projet.

Pour y répondre, j'ai développé le module `health_test.c`. Il tourne en tâche de fond et surveille continuellement le flux sortant. Si le ratio de bits à 1 ou 0 s'écarte d'un seuil critique que j'ai défini, le système passe immédiatement en état d'alerte.

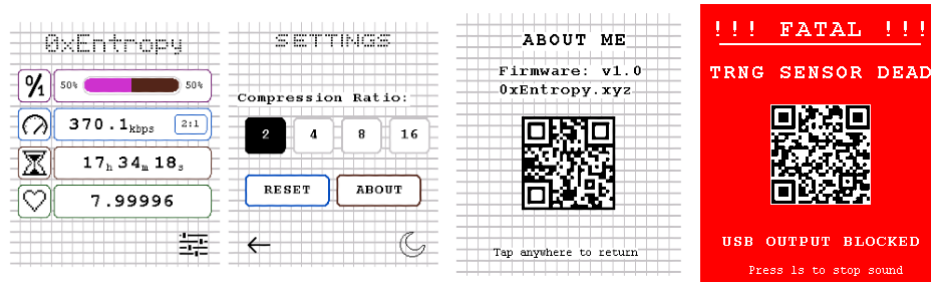


Figure 11: Maquette de l'interface graphique (Dashboard) permettant le suivi en direct (Couleur inversée pour impression)

Justification de mes choix :

Au lieu de dépendre d'un logiciel d'analyse à installer sur l'ordinateur hôte, j'ai voulu rendre mon prototype totalement autonome. J'ai donc choisi d'exploiter un écran tactile LCD intégré directement sur le boîtier. La maquette (11) présente le tableau de bord (*Dashboard*) que j'ai imaginé et codé pour l'appareil.

J'ai programmé cette interface pour qu'elle affiche les métriques de santé essentielles en temps réel :

1. **L'équilibre des bits** (ratio 0/1, avec une cible idéale à 50%).
2. **Le débit de génération** (en octets/seconde).
3. **Le temps de travail** (Temps écoulé depuis le démarrage).
4. **L'entropie de Shannon calculée** (avec pour objectif d'atteindre ~7.9999 bits/octet).
5. **Le ratio de compression SHA-256** : paramètre que j'ai rendu modifiable dynamiquement depuis le menu des réglages (*Settings*).

L'ajout de ce réglage du ratio SHA-256 directement dans l'interface est une fonctionnalité que je tenais particulièrement à implémenter. Concrètement, ce ratio définit combien de bits bruts sont "avalés" par ma fonction de hachage pour produire les bits finaux. Laisser ce choix accessible permet d'expérimenter et d'adapter le générateur selon le besoin : un ratio faible augmente mon débit de génération (vitesse) mais offre un blanchiment statistique plus léger, tandis qu'un ratio élevé garantit une entropie maximale au détriment de la vitesse.

10 Cas pratique et résolution des défis techniques

Dans ce chapitre, je présente la phase de réalisation concrète de mon circuit. De la première maquette sur la breadboard jusqu'à la conception de mon propre circuit imprimé, en passant par des sessions de débogage logiciel complexes, je détaille ici les obstacles techniques que j'ai rencontrés sur le terrain et les solutions que j'ai mises en œuvre pour atteindre mes objectifs.

10.1 Évolution matérielle : De la breadboard au circuit imprimé

La partie matérielle du projet a été réalisée en plusieurs étapes. Chaque nouveau montage m'a permis de corriger les erreurs du précédent.

10.1.1 Les premiers montages et l'identification du bruit

Mon tout premier montage était très rudimentaire. Il consistait simplement en un amplificateur opérationnel, une diode Zener de 7V et un convertisseur Boost. Ce premier essai sur platine d'essai (breadboard) s'est soldé par un échec : je n'obtenais aucun signal chaotique exploitable. Plusieurs raisons expliquent ce résultat. D'une part, la tension de claquage de la diode Zener (7V) était trop faible pour générer un bruit d'avalanche suffisant. D'autre part, un problème de conception avec la résistance de tirage (pull-up) en sortie forçait constamment le signal à l'état haut (1).

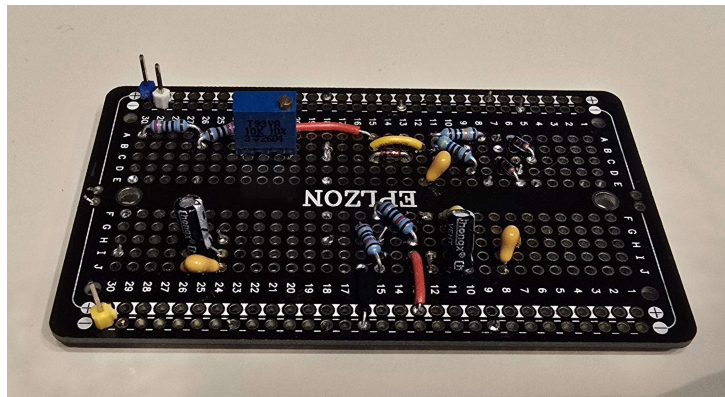


Figure 12: Premier prototype sur carte de prototypage)

À cause d'un retard de livraison de matériel (grève postale), j'ai dû patienter avant de réaliser un second montage avec de nouveaux composants. J'ai mis ce temps à profit pour commencer le développement logiciel, en l'occurrence la gestion de l'écran et du haut-parleur, mais aussi pour concevoir le script côté PC chargé de récupérer les bits générés.

Une fois le matériel reçu, j'ai réalisé ce deuxième essai et obtenu mes premiers résultats concluants. J'ai soumis ce signal brut aux outils d'évaluation de l'entropie du NIST (notamment `ea_non_iid`). Comme le précise le NIST, l'objectif de cet outil n'est pas d'évaluer la qualité cryptographique du flux de sortie, mais de mesurer l'entropie de la source. Cela m'a permis de prouver que mon signal contenait bien de l'aléa véritable et que mon phénomène physique fonctionnait correctement.

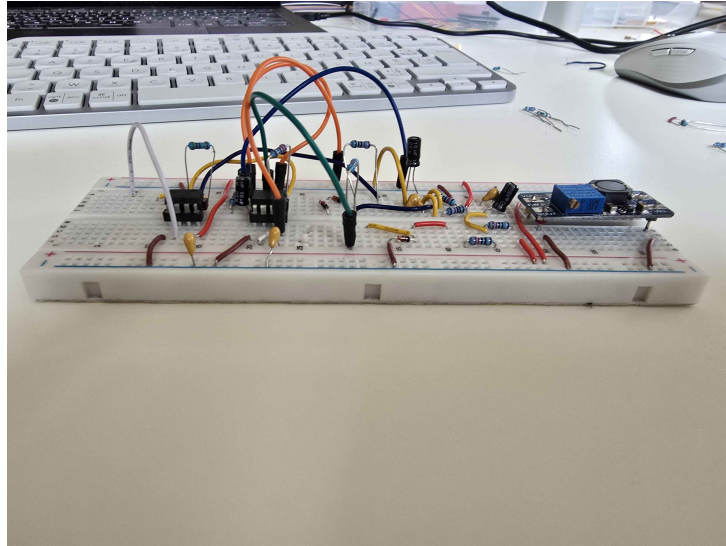


Figure 13: Deuxième prototype sur breadboard

Justification des choix de conception : Ce montage me donnait un débit d'environ 300 bits/s, mais il était extrêmement instable. Le moindre mouvement d'un câble faisait chuter l'entropie. C'est à ce moment que j'ai pris la décision de retirer les diodes d'écrêtage (*clamping*). En observant le comportement du circuit, j'ai remarqué que les légères variations de qualité de ces composants provoquaient des fuites du signal. Changer les diodes changeait drastiquement les résultats. Étant donné que mon condensateur de liaison (post-diode Zener) bloquait déjà le 12V continu, le risque de griller l'ampli était très faible. J'ai donc préféré simplifier le circuit et retirer le *clamping* pour préserver l'intégrité du bruit. Ce montage datait d'un design antérieur.

10.1.2 Les limites de la breadboard et l'amplificateur inverseur

Pour tenter d'augmenter mon débit, j'ai essayé de reproduire ce montage de manière plus dense avec des câbles DuPont. Ce sont des fils pour breadboard avec des pins en métal larges se bloquant mieux dans la breadboard. Le circuit est vite devenu un enfer de fils. Même si j'ai réussi à monter à 1600 bits/s brut, le système était capricieux : dès que j'augmentais la vitesse de lecture sur le microcontrôleur, la qualité du signal s'effondrait.

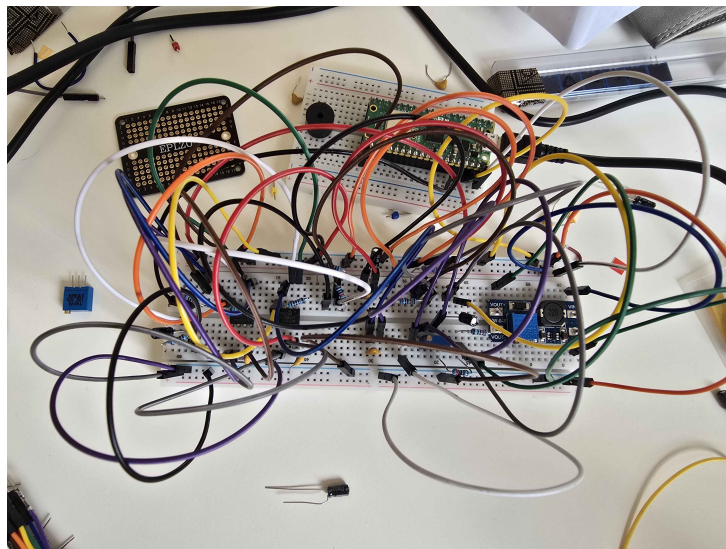


Figure 14: Troisième prototype sur breadboard

J'ai alors identifié un problème de conception majeur. En retirant la diode Zener du circuit, un bruit

résiduel persistait. Ce bruit avait un score d'entropie très médiocre. Je souhaitais donc enlever cette constante. Aussi, plus je m'approchais de la breadboard avec ma main, plus le signal fluctuait. Après quelques tests, j'ai remarqué que l'amplificateur lui-même générait du bruit lorsqu'il était alimenté par ma tension de référence (V_{ref}).

Solution apportée : Pour en avoir le cœur net, j'ai modifié l'architecture de mon circuit en passant sur un montage non-inverseur. J'ai arrêté de fournir V_{ref} à l'amplificateur et je l'ai relié à la masse. Le résultat a été immédiat : sans la Zener, je n'avais plus aucun bruit parasite (à l'exception de quelques faux contacts quand je bougeais la plaque ou quand j'approchais ma main).

10.1.3 Le passage au circuit soudé

Pour me débarrasser définitivement des capacités parasites des câbles de la breadboard, j'ai décidé de souder exactement le même circuit sur une carte de prototypage (*perfboard*).

Cette étape a été un grand bond en avant. Sans même modifier la vitesse de rafraîchissement dans mon code, mon débit a bondi à 10 000 bits/s ! En réalité, la breadboard manquait de précision et de fiabilité pour un montage aussi sensible. Le simple fait de passer sur un circuit soudé a résolu ces problèmes de perturbation et m'a permis d'obtenir enfin un signal propre et exploitable.

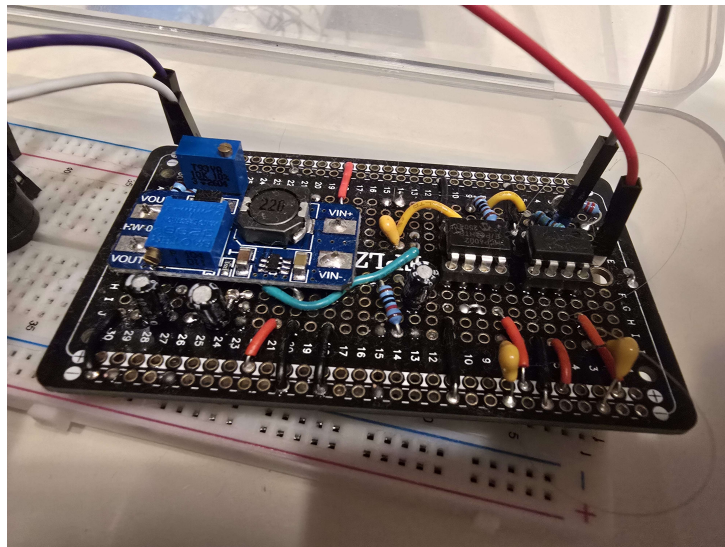


Figure 15: Carte de prototypage soudée (Perfboard)

C'est le prototype où tout le code a été développé. Une solution définitive a été créée dans une boîte en plastique où des trous ont été percés pour l'écran et le câble.

10.1.4 L'aboutissement : Le PCB sur mesure

Même si la perfboard fonctionnait très bien, elle restait fragile et encombrante. Pour répondre à mon exigence de taille (*User Story* définie au début du projet) et rendre mon TRNG reproductible, j'ai conçu un circuit imprimé (PCB) sur mesure avec KiCad.

J'ai repris mon schéma et ajouté une deuxième source d'entropie. Le comparateur et l'amplificateur sont de type double. Ce qui signifie que sur une seule puce, il y a la capacité d'amplifier et de comparer deux circuits totalement indépendamment. Il est donc peu coûteux en place de rajouter une deuxième voie. Mon débit a atteint +/- 900 000 bits/s. Mais le routage physique de la carte m'a confronté à de nouveaux défis, notamment la cohabitation entre mon signal analogique ultra-sensible et les signaux numériques haute fréquence.

Justification des choix de routage :

Isolation du bruit :

J'ai utilisé des plans de masse (couches de cuivre reliées à la terre) sur tout le circuit pour créer un blindage électromagnétique. J'ai aussi séparé physiquement au maximum le bloc analogique du bloc numérique.

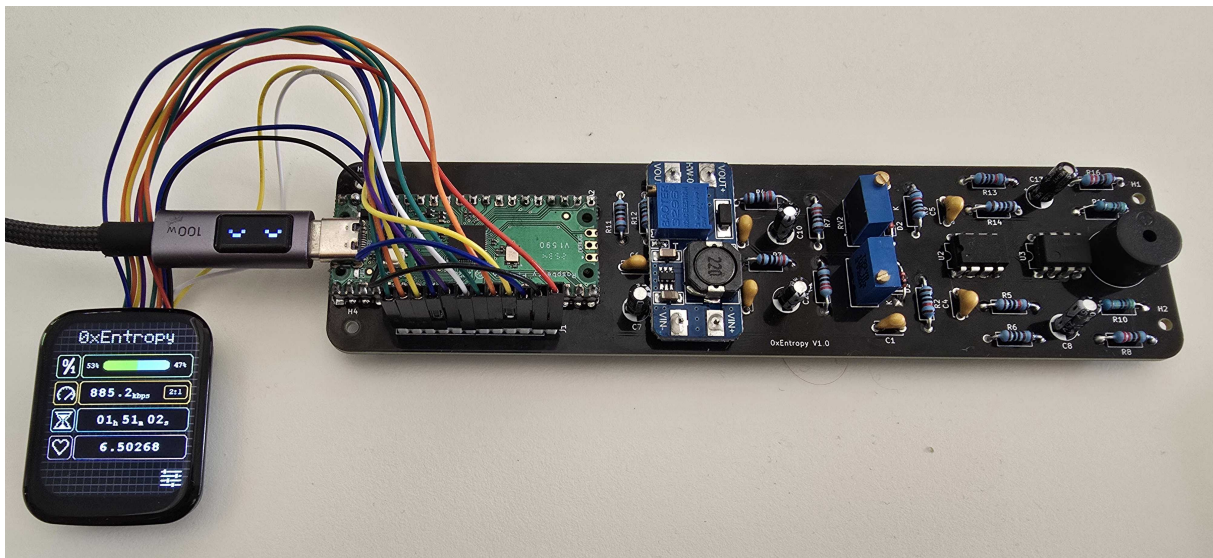


Figure 16: Photographie du circuit imprimé (PCB) finalisé et fonctionnel, avec l'ensemble des composants soudés.

10.2 Défis logiciels

C'est à partir de la version soudée que j'ai entièrement réécrit mon code en C. Comme expliqué dans le chapitre sur la conception, j'ai pu mettre en place l'échantillonnage matériel (PIO) et la séparation sur les deux cœurs du processeur, ce qui m'a permis d'atteindre mes objectifs de vitesse sans brider la récupération de l'aléa.

Cependant, la partie logicielle m'a réservé une des pannes les plus difficiles à diagnostiquer du projet.

10.2.1 Le mystère de l'octet 13

Pendant mes tests, j'ai remarqué une anomalie illogique. Je générais un fichier de données sur le Pico, je le blanchissais sur mon PC, et j'obtenais un score d'entropie parfait de 7.999. Mais si je demandais au Pico de faire lui-même le blanchiment puis de m'envoyer le fichier, le score chutait à 7.81, sans aucune raison apparente.

J'ai passé deux jours entiers sur ce problème. J'ai fini par afficher le flux en hexadécimal sur mon écran et je l'ai regardé défiler. J'ai alors remarqué une anomalie visuelle : les lignes de texte ne revenaient pas à la ligne à la même vitesse.

L'explication technique : En faisant des recherches sur le comportement du microcontrôleur, j'ai découvert que le coupable était le kit de développement standard (SDK). Par défaut, pour formater correctement du texte dans un terminal externe, le SDK intercepte chaque octet de valeur 10 (le saut de ligne `\n`) et ajoute automatiquement un octet 13 (le retour chariot `\r`) juste avant. Dans un flux aléatoire brut, la valeur 10 apparaît très régulièrement. Le SDK de mon microcontrôleur modifiait donc silencieusement mon flux en y injectant un motif répétitif, ce qui détruisait mon entropie au moment de l'envoi vers le PC.

Solution apportée : Il m'a suffi d'ajouter une ligne de configuration (`stdio_set_translate_crlf(0)`) pour interdire au SDK d'altérer mes données. Dès lors, j'ai retrouvé des résultats parfaits en sortie USB.

10.3 Interface graphique : Esthétisme et contraintes de mémoire

Comme défini dans mes objectifs, l'appareil devait être autonome et afficher ses propres statistiques (*health tests*). Cependant, le rôle principal du microcontrôleur est de produire de l'entropie. Je ne pouvais pas me permettre de saturer sa mémoire ou son processeur avec des bibliothèques graphiques lourdes.

Justification des choix techniques :

Pour garder une interface légère, j'ai opté pour une approche bas niveau. J'ai dessiné mes logos et mes menus en *pixel art* sur un outil externe, puis je les ai exportés sous forme de tableaux *Bitmap* monochromes.

Dans un fichier Bitmap de ce type, l'information est stockée de la manière la plus brute possible : 1 bit de mémoire correspond exactement à 1 pixel de l'écran (0 pour éteint, 1 pour allumé). J'ai ensuite codé en C ma propre fonction de dessin capable de lire ces tableaux binaires et d'allumer les bons pixels sur l'écran LCD. Cette technique m'a permis d'obtenir une interface visuellement riche et claire pour l'utilisateur (voir Figure 11), tout en conservant une empreinte mémoire extrêmement faible, laissant ainsi toutes les ressources disponibles pour la cryptographie.

11 Stratégie de validation et résultats statistiques

La validation de mon générateur d'aléa matériel repose sur une stratégie à deux niveaux : une validation physique (matérielle) pour garantir l'origine de l'entropie, et une validation statistique (logicielle) pour certifier la qualité mathématique des nombres générés.

11.1 Validation matérielle : Contrôle de l'origine du bruit

Comme je l'ai détaillé dans le chapitre précédent, il était crucial pour moi de m'assurer que l'entropie provenait bien de ma source physique (le bruit d'avalanche) et non de bruits parasites liés à l'alimentation ou à l'environnement. Pour chaque montage, j'ai réalisé une campagne de tests en retirant volontairement la source de bruit (la diode). Mon objectif lors de cette manipulation était d'observer l'effondrement de la génération de bits aléatoires. J'ai ainsi pu prouver que mon circuit ne numérise pas du "vide" ou des interférences, et que la source d'entropie est bien isolée et maîtrisée.

11.2 Validation logicielle : L'outil `ent` et l'importance du volume de données

Pour évaluer la qualité des données que j'ai générées, j'ai utilisé l'outil d'analyse statistique `ent` sur mes différents prototypes. En complément, et comme je l'ai évoqué plus tôt dans ce document, j'ai également fait appel à l'outil `ea_non_iid` (issu de la suite NIST) pour estimer l'entropie brute (min-entropie) de ma source.

Dans un premier temps, j'ai remarqué que les résultats semblaient sensiblement identiques entre les prototypes. Cependant, je me suis rapidement heurté à une limite majeure commune à ces logiciels d'analyse : le débit très faible de mes premiers circuits. Générer à peine 100 ko ou 3 Mo me prenait des journées entières. Or, j'ai compris que les tests statistiques (tout comme l'évaluation de la min-entropie par `ea_non_iid`) requièrent un "historique" conséquent pour être précis. Si je teste un fichier de seulement 100 ko, même avec un excellent générateur, les résultats seront moyens car l'échantillon est trop petit pour lisser les probabilités. J'ai réalisé qu'il me fallait au minimum 1 Mo de données, idéalement beaucoup plus, pour obtenir une estimation statistique fiable. Ma comparaison entre les premiers prototypes était donc biaisée par ce manque de données. Limité par le débit, impossible de comparer les prototypes de manière cohérente sans avoir besoin de mois entiers de récupération de données.

Grâce à l'amélioration du débit sur mon Prototype 3, j'ai pu récolter un fichier massif de plus de 6 Go. Les résultats que j'ai obtenus avec l'outil `ent` sur ce fichier final frôlent la perfection théorique, comme je le résume dans le tableau ci-dessous :

Paramètre évalué par <code>ent</code>	Résultat mesuré (Prototype 3)	Interprétation / Valeur idéale
Entropie	8.000000 bits par octet	Idéal absolu (8.0). Aucune compression possible (0%).
Test du Khi carré	194.60 (dépassé dans 99.81 % des cas)	Indique une distribution parfaitement uniforme des bits.
Moyenne arithmétique	127.4996	Très proche de la perfection théorique de 127.5.
Estimation de Pi (Monte Carlo)	3.141575582	Erreur de 0.00 %. Prouve une répartition spatiale aléatoire.
Coefficient de corrélation sérielle	0.000009	Quasiment 0.0. Prouve l'absence de prédictibilité linéaire.

Table 6: Résultats de l'analyse statistique avec l'outil `ent` sur le Prototype 3

11.3 Tests avancés : La suite Dieharder (Tests 1 à 17)

Pour aller plus loin que les statistiques de base, j'ai soumis mon fichier de 6 Go à la rigoureuse suite de tests cryptographiques **Dieharder**.

Tous les tests que j'ai sélectionnés ont obtenu la note **PASSED** (je détaille ces épreuves dans l'**Annexe B**). Il est important de préciser que j'ai exécuté la suite Dieharder uniquement sur les tests 1 à 17, et non dans son entièreté.

Justification de ce choix : J'ai fait ce choix car les tests ultérieurs de la suite exigent des volumes de données colossaux (plusieurs dizaines, voire centaines de gigaoctets). Bien que j'aie grandement amélioré le débit de mon prototype 3, récupérer de manière ininterrompue une telle quantité d'aléa pur m'aurait pris un temps démesuré. Les 17 premiers tests couvrent déjà les failles les plus complexes et me suffisent amplement à valider la robustesse de mon système dans le cadre de ce TFE.

11.4 Pistes d'amélioration et perspectives futures

Bien que mon prototype final ait démontré son efficacité et validé les hypothèses de mon travail, ce projet ouvre la voie à plusieurs évolutions techniques et pratiques. Dans l'optique de perfectionner ce générateur d'aléa, voici les principales pistes d'amélioration que j'envisage pour une future itération :

Miniaturisation avec des composants CMS :

Actuellement, le prototype utilise des composants traversants, ce qui facilite le prototypage manuel mais occupe un espace important. Le passage à une conception basée sur des composants montés en surface permettrait de réduire drastiquement la taille du circuit. Outre l'avantage esthétique et compact, raccourcir les pistes électroniques permettrait de limiter la captation des bruits parasites environnants, améliorant ainsi l'intégrité du signal analogique sensible.

Conception d'un boîtier imprimé en 3D :

Afin de protéger le matériel des chocs, de la poussière et des courts-circuits accidentels lors des manipulations, la conception d'un boîtier sur mesure en impression 3D serait une suite logique. Cela donnerait au projet un véritable aspect de produit fini et portable, prêt à être branché en toute sécurité comme une clé USB classique.

Création d'une page web d'explication pédagogique :

Dans une démarche d'ouverture et de partage de connaissances, je souhaiterais développer une interface web documentant le projet. Cette page servirait à vulgariser le fonctionnement interne du générateur. Elle permettrait d'expliquer au grand public et aux passionnés comment un simple bruit d'avalanche physique est capté, amplifié, puis transformé en une véritable suite de bits imprévisibles.

Accumulation de données pour la suite de tests *Diehard* :

Comme je l'ai expliqué dans mon analyse des limites logicielles, la validation statistique absolue demande des volumes de données gigantesques. Une amélioration majeure de la phase de test consisterait à laisser tourner le générateur de manière ininterrompue sur une longue période (plusieurs semaines si nécessaire). L'objectif serait d'accumuler suffisamment de gigaoctets d'aléa pur pour pouvoir soumettre le fichier à l'intégralité de la célèbre suite de tests *Diehard* (ou son évolution *Dieharder*), offrant ainsi une certification mathématique incontestable et exhaustive.

12 Contraintes législatives et normatives

La conception d'un dispositif matériel électronique est soumise à un cadre légal et normatif strict. Bien que le projet « 0xEntropy v1.0 » soit actuellement à l'état de prototype académique, son développement a été pensé en tenant compte des directives européennes et internationales.

Voici l'analyse des contraintes pertinentes et leur intégration dans la réalisation du projet :

12.0.1 Directive RoHS (Restriction of Hazardous Substances)

- **La contrainte :** La directive européenne 2011/65/UE (RoHS) [14] limite l'utilisation de certaines substances dangereuses (plomb, mercure, cadmium, etc.) dans les équipements électriques. Tout produit vendu en Europe doit s'y conformer.
- **Prise en compte dans la réalisation :**
 - **Sélection des composants :** Tous les éléments choisis (Raspberry Pi Pico, amplificateurs MCP6002, comparateurs LM393) disposent d'une certification RoHS stricte fournie par leurs fabricants.
 - **Assemblage matériel :** Lors de la soudure du circuit imprimé (PCB) final, j'ai veillé à utiliser exclusivement de la **soudure sans plomb**, respectant ainsi la santé de l'utilisateur et les normes environnementales.

12.0.2 Directive DEEE / WEEE (Gestion des déchets électroniques)

- **La contrainte :** La directive européenne 2012/19/UE (DEEE) [15] impose aux concepteurs de prévoir et de faciliter le recyclage des appareils en fin de vie pour limiter la pollution électronique.
- **Prise en compte dans la réalisation :**
 - **Conception modulaire :** Les composants les plus complexes (Raspberry Pi Pico, écran LCD) ne sont pas soudés de manière permanente au circuit de génération d'avalanche (utilisation de connecteurs de type *headers* ou de câblages détachables).
 - **Éco-conception en fin de vie :** Cette modularité permet un démontage facile pour le recyclage ou la réutilisation des composants majeurs, respectant pleinement les principes de la directive.

13 Sécurité et intégrité du système

La sécurité est le but premier de mon Générateur de Nombres Aléatoires (TRNG). Si le flux d'entropie est compromis ou biaisé, c'est l'ensemble de la chaîne cryptographique qui en dépend qui s'effondre. J'ai donc veillé à ce que la sécurité ne se limite pas au produit final : je l'ai intégrée « *by design* » (dès la conception) à toutes les étapes de mon projet, incluant la protection du flux de données, la sécurisation de mon processus de développement et la maintenance à long terme de l'appareil.

Ce chapitre identifie les éléments critiques de mon prototype « 0xEntropy v1.0 », les risques qui s'y appliquent, et détaille les contre-mesures que j'ai mises en place pour les neutraliser.

13.0.1 Protection de la source d'entropie (Attaques physiques)

Élément à protéger :

Le signal analogique (bruit d'avalanche) et le flux binaire brut qui en découle.

Risque identifié :

L'attaque par injection de fréquence (*Frequency Injection Attack*) et la défaillance matérielle silencieuse.

Lors de l'analyse des risques, j'ai identifié l'attaque par injection de fréquence comme une menace majeure, comme démontré par Markettos et Moore dans leurs travaux sur la vulnérabilité des générateurs d'aléa. [5] Elle consiste, pour un attaquant externe, à perturber le circuit (via des ondes électromagnétiques ou des manipulations de l'alimentation) pour forcer le bruit d'avalanche à osciller à une fréquence fixe et connue. Le signal perdrait alors sa nature quantique et deviendrait déterministe. De plus, un composant électronique peut simplement griller, figeant la sortie sur un état continu (que des 0 ou que des 1).

Justification de mes choix :

Pour contrer ces risques critiques, j'ai implémenté une sécurité algorithmique agissant comme un « coupe-circuit » matériel (*fail-safe*). J'exploite le module de tests de santé (`health_test.c`) détaillé dans la conception logicielle. Ce moniteur que j'ai codé surveille l'entropie en continu. Si l'écart de variance chute drastiquement (absence de bruit) ou qu'un motif répétitif caractéristique d'une injection de fréquence est détecté, le système réagit immédiatement :

1. Coupure du débit :

Je bloque instantanément le flux de données vers l'utilisateur pour éviter toute fuite de nombres non aléatoires.

2. Alerte physique :

Le système bascule l'interface utilisateur (écran) en mode alerte critique et déclenche le buzzer (BZ1) pour avertir physiquement l'utilisateur de la compromission du système.

Cette contre-mesure garantit que mon appareil préférera toujours s'arrêter de fonctionner plutôt que de délivrer des données faillibles.

13.0.2 Maintenance et pérennité à long terme de la solution

Élément à protéger :

La viabilité et la fiabilité de mon TRNG dans le temps.

Risque identifié :

Vieillesse des composants analogiques (dérive thermique, usure de la diode Zener) et découverte future de failles dans les algorithmes de hachage.

Justification de mes choix :

J'ai conçu ce matériel pour qu'il puisse être maintenu et garantir sa sécurité sur plusieurs années. Pour anticiper le vieillissement physique du cœur d'avalanche (qui pourrait faire baisser l'amplitude du bruit avec les années), j'ai intégré un **potentiomètre de calibration** (RV1) dans le circuit matériel. Il permet de réajuster manuellement le point de polarisation de la diode lors d'opérations de maintenance, sans avoir à dessouder de composants sur mon circuit imprimé.

Côté logiciel, j'ai pris en compte le fait que la sécurité à long terme exige de pouvoir appliquer des mises à jour (*patches*). Si l'algorithme SHA-256 venait à présenter des vulnérabilités dans le futur, ou si le firmware nécessitait une correction, mon choix d'utiliser le Raspberry Pi Pico permet une mise à jour extrêmement simple et sécurisée. Il suffit de connecter le dispositif en USB et de glisser-déposer le nouveau fichier binaire compilé (`.uf2`), garantissant ainsi que mon appareil peut évoluer face aux menaces futures. Enfin, j'ai archivé l'intégralité de ma nomenclature (BOM) et de mes fichiers de fabrication (Gerber), assurant ainsi la reproductibilité et la maintenance matérielle de ma solution à très long terme.

14 Conclusion

Les ordinateurs sont conçus pour être parfaitement prévisibles, ce qui est génial pour faire tourner nos programmes, mais terrible quand on a besoin d'aléatoire pur pour la cryptographie. C'est exactement ce paradoxe qui m'a fasciné au début de ce TFE : comment forcer une machine ultra-logique à produire du vrai chaos ? C'est ce défi qui a donné naissance à **0xEntropy**.

14.0.1 Bilan : du concept physique au besoin du client

Au départ, la demande de Julien Castiaux était ambitieuse : il voulait un vrai générateur matériel (TRNG) qui exploite un phénomène physique, qui tienne sur un bureau, qui s'alimente en USB et qui recrache des données prêtes à être exploitées par un développeur.

Aujourd'hui, je peux dire que le contrat est rempli. On est passé d'une simple idée théorique d'un dispositif portable à un appareil fonctionnel. Mais là où les retours de Julien ont été vraiment cruciaux, c'est sur l'aspect "produit fini". En tant que développeur, il m'a fait comprendre qu'une bonne source de bruit facilement reproductible n'est pas disponible en open hardware ou sans calibration précise. C'est ce qui m'a poussé à aller plus loin que la simple maquette électronique en intégrant un écran un comparateur et amplificateur fonctionnant de pair, en plus des *health tests* en temps réel et un algorithme de hachage (SHA-256) directement dans le Raspberry Pi Pico.

14.0.2 Ce que j'en retiens : réalisation et méthode

Si je dois faire un bilan critique de ce projet, je dirais que le plus grand choc a été la confrontation entre la théorie et la pratique. L'analogique, c'est un monde à part. Essayer de dompter un signal de quelques millivolts noyé dans le bruit de l'alimentation USB sur une *breadboard* capricieuse a été un vrai casse-tête. J'ai vraiment pris conscience que le passage au circuit imprimé soudé n'était pas juste là pour faire joli : c'était vital pour l'intégrité du signal.

14.0.3 Comment faire vivre ce projet demain ?

Je ne voulais pas que 0xEntropy finisse dans un tiroir une fois mon diplôme en poche. L'électronique vieillit, et le comportement de la diode Zener va forcément dériver avec le temps et la température. C'est pour ça que j'ai intégré un potentiomètre permettant une calibration manuelle, assurant la maintenance du système sur le long terme sans devoir tout redessiner. Je compte aussi développer un logiciel très visuel et ludique et laisser mon dispositif tourner sans arrêt. Ainsi, je verrai ce que provoque l'usure sur mon système avec le temps.

De plus, l'utilisation du Raspberry Pi Pico rend les mises à jour logicielles enfantines : un simple glisser-déposer d'un fichier `.uf2` suffit pour patcher le firmware si une faille de sécurité est découverte dans le futur. Enfin, la publication de mes schémas et de mon code en *open source* garantit que n'importe qui pourra reprendre, auditer, réparer ce projet proprement.

14.0.4 Et pour la suite ?

Même si cette version 1.0 valide totalement mes hypothèses, j'ai déjà plein d'idées pour la suite. La prochaine étape logique, ce serait de passer sur des composants CMS (montés en surface). Ça permettrait de miniaturiser la carte au point d'en faire une simple clé USB. Je pourrais aussi explorer d'autres puces pour booster la vitesse de transfert USB et augmenter drastiquement le débit de génération.

En bref, ce TFE a été une aventure technique incroyable. Il m'a poussé à sortir de ma zone de confort en mêlant électronique analogique, développement bas niveau et sécurité. 0xEntropy est la preuve concrète qu'avec des composants abordables et beaucoup de bidouille méthodique, on peut capturer un morceau de l'univers physique pour sécuriser notre monde numérique.

Bibliographie

- [1] Cloudflare, Inc. *Randomness 101: LavaRand in Production (Lava Lamp Encryption)*. n.d. URL: <https://www.cloudflare.com/learning/ssl/lava-lamp-encryption/> (visited on 05/07/2026).
- [2] Gabriel Guerrero. *RAVA: an Open Hardware True Random Number Generator based on Avalanche Noise*. https://github.com/gabrielguerrer/rng_rava. Accessed: 2026-04-28. 2026.
- [3] *Make your own True Random Number Generator 2*. URL: <http://robseward.com/misc/RNG2/> (visited on 04/27/2026).
- [4] Nicolas Bochart et al. “True-randomness and pseudo-randomness in ring oscillator-based true random number generators”. In: *International Journal of Reconfigurable Computing* 2010 (2010), pp. 1–13.
- [5] A. Theodore Marketos and Simon W. Moore. “The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators”. In: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 317–331. DOI: 10.1007/978-3-642-04138-9_23.
- [6] Wolfgang Killmann and Werner Schindler. *A proposal for: Functionality classes for random number generators*. AIS 31, Version 2.0. Adopted by the European SOG-IS agreement for Common Criteria evaluations. Bundesamt für Sicherheit in der Informationstechnik (BSI), 2011.
- [7] National Institute of Standards and Technology. *Recommendation for the Entropy Sources Used for Random Bit Generation*. NIST Special Publication 800-90B. 2018.
- [8] David Johnston. *Random Number Generators: Principles and Practices*. De Gruyter, 2018. ISBN: 978-1501515132. DOI: 10.1515/9781501506062.
- [9] Robert G. Brown. *Dieharder: A Random Number Test Suite*. Software suite for testing random number generators. 2004.
- [10] *DIGITAL INTERFACE CIRCUIT FOR A RANDOM NOISE GENERATOR*. June 1974. URL: <https://www.freepatentsonline.com/3816765.html>.
- [11] Kevin Ginunas Mengesha Tekle Kertikey Thakar. “Zener Diodes Introduction”. In: (Oct. 2025), p. 3. URL: <https://www.ti.com/lit/an/slvag00/slvag00.pdf>.
- [12] TDK Corporation. *Capacitors (Tech Magazine)*. n.d. URL: <https://www.tdk.com/en/tech-mag/capacitor/03> (visited on 05/05/2026).
- [13] Deesdee. *0xEntropy : Code source et schémas du projet*. <https://git.plpi.xyz/Deesdee/0xEntropy>. Dépôt Git officiel du projet. 2026.
- [14] Union européenne. *Directive 2011/65/UE relative à la limitation de l’utilisation de certaines substances dangereuses dans les équipements électriques et électroniques (RoHS)*. Journal officiel de l’Union européenne, L 174. 2011. URL: <https://eur-lex.europa.eu/legal-content/FR/TXT/?uri=CELEX%3A32011L0065> (visited on 05/28/2026).
- [15] Union européenne. *Directive 2012/19/UE relative aux déchets d’équipements électriques et électroniques (DEEE)*. Journal officiel de l’Union européenne, L 197. 2012. URL: <https://eur-lex.europa.eu/legal-content/FR/TXT/?uri=CELEX%3A32012L0019> (visited on 05/28/2026).