Geradores de Números Aleatórios

Grupo 8:

Ariel Yssou	8944644
Luís E. Previdente	7994937
Marcus Lemes	7994663

Trabalho MEFE 2017-2

Hipótese Nula $\mathcal{H}_0$

- Hipótese de que não existe relação entre dois fenômenos medidos – ou dois conjuntos de dados, ou qualquer hipótese alternativa que negue a hipótese do experimento analisado.

- Geralmente, em Física, o objetivo dos experimentos é descartar a hipótese nula.

Hipótese Nula $\mathcal{H}_0$

No caso deste trabalho, aplicado a geradores aleatórios, descartar a hipótese nula significa encontrar um problema no gerador (assumindo que o teste seja bom).

- $\mathcal{H}_0$ = "Meu gerador é lindo e perfeito"
- $\mathcal{H}_a$ = "Talvez nem tanto"

Quando $\mathcal{H}_0$ é válida?

- Nenhum gerador computacional é perfeito. Dizer que a hipótese nula é verdadeira, neste caso, é uma abreviação para "é razoável assumir que os números gerados são aleatórios para fins práticos". Para ver se podemos aceitá-la, usamos testes estatísticos.

Um exemplo

Para testar uma simulação de moeda, podemos contar o número de caras de um conjunto de jogadas. Assumindo que a moeda é "perfeita" (hipótese nula), deveríamos esperar uma distribuição binomial de parâmetro meio.

Valor-p

Usando a distribuição cumulativa, podemos calcular a probabilidade do conjunto de dados gerado sob a hipótese nula $\mathcal{P}(\ D\ |\mathcal{H}_0)$. Este valor é chamado de $valor - p$.

Significado do valor-p

- O valor-p não é a probabilidade da hipótese nula ser verdadeira. Valor-p é a probabilidade de um gerador "verdadeiramente aleatório" resultar no conjunto de dados obtido.

- $\mathcal{P}(\mathcal{H}_0|\, D) \neq \mathcal{P}(D|\mathcal{H}_0)$

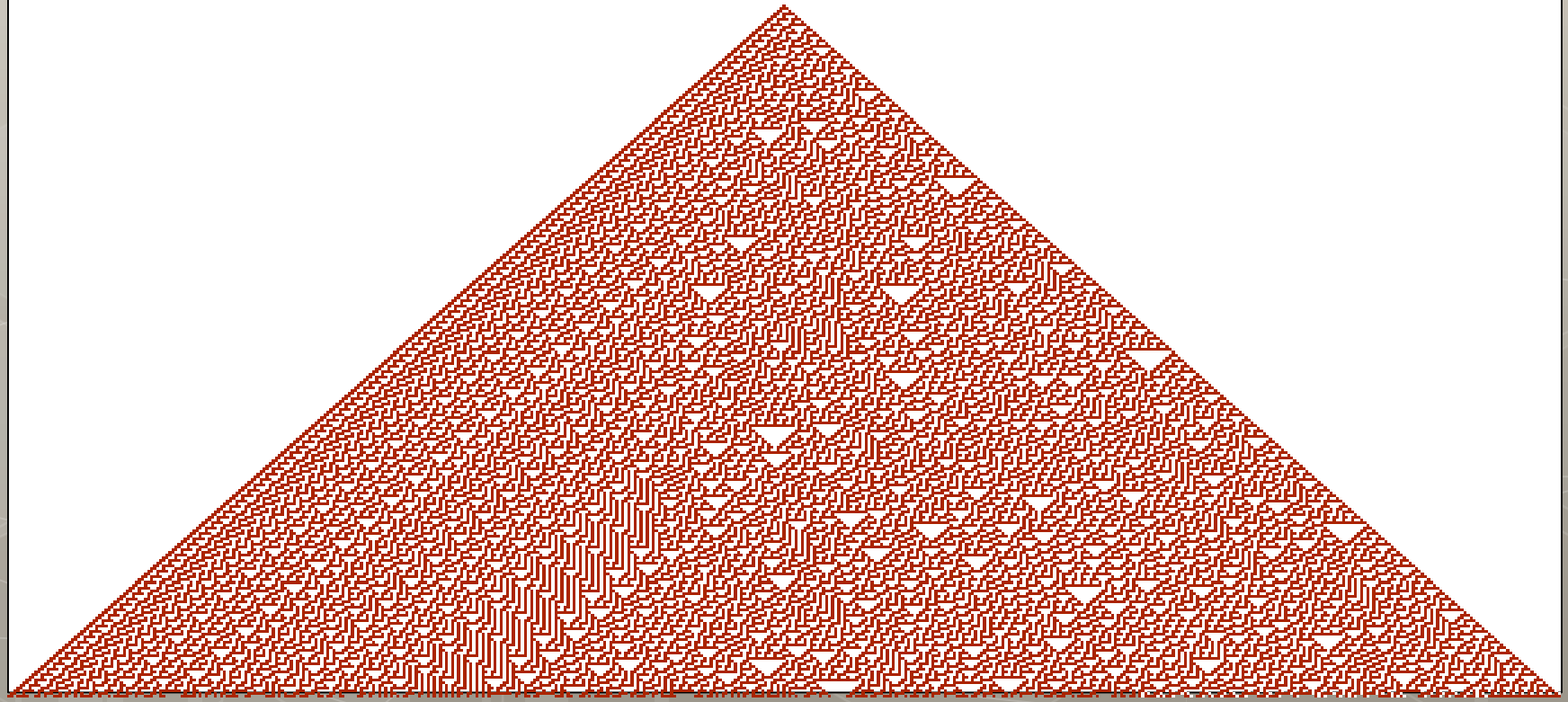
Testes Estatísticos

- Aplicamos os geradores a situações onde números aleatórios deveriam possuir certas estatísticas.

Alguns geradores

- Mersenne Twister: É o padrão para o Matlab/Octave e para Phyton e é o system rand de algumas distribuições de Linux. Já foi o mais rápido de todos, hoje é considerado lento para uso em simulações extensivas.

Extended Cellular Automaton Generator (Mathematica)



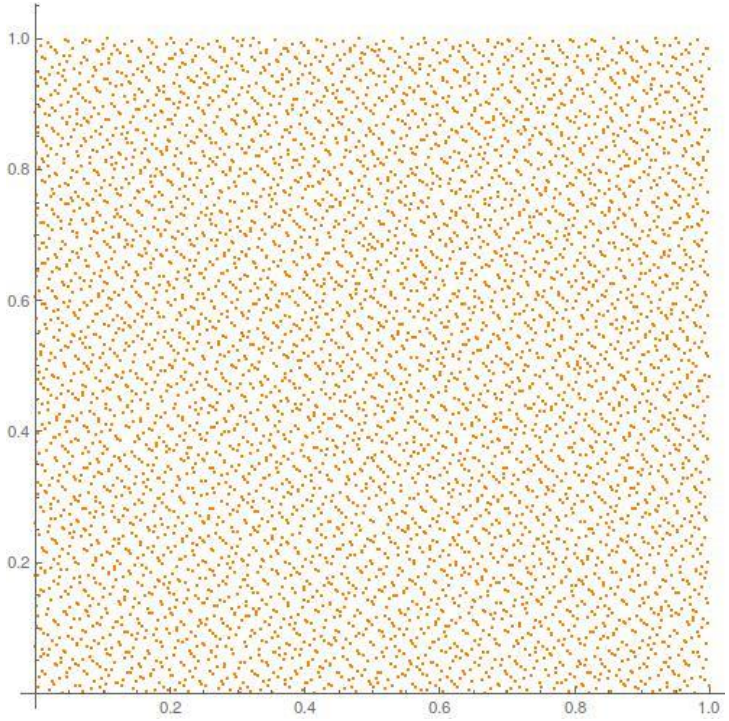
Extended Cellular Automaton Generator (Mathematica)

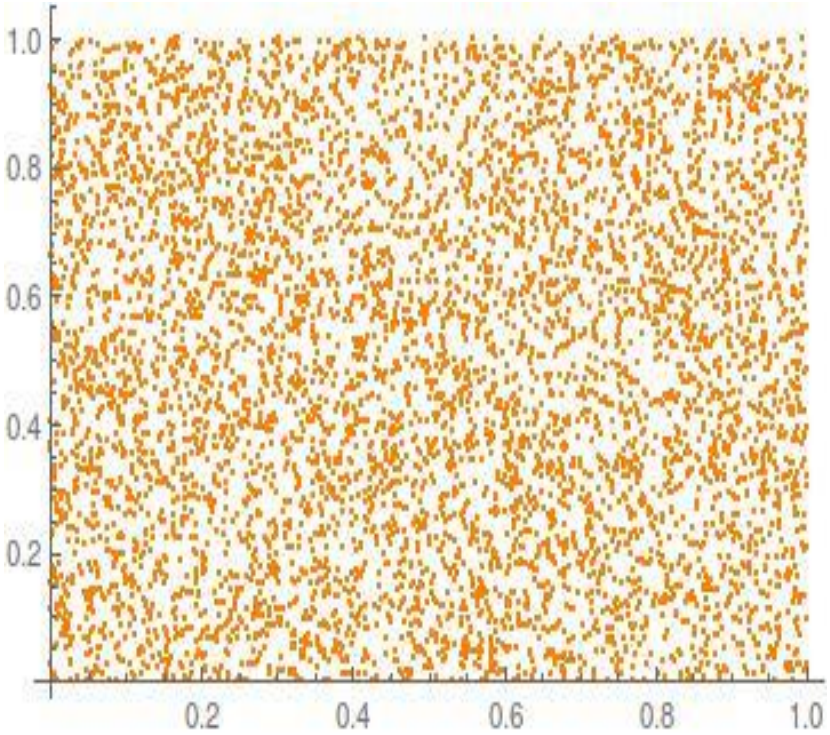
Extended Cellular Automaton Generator (Mathematica)

“Quick and dirty generators”

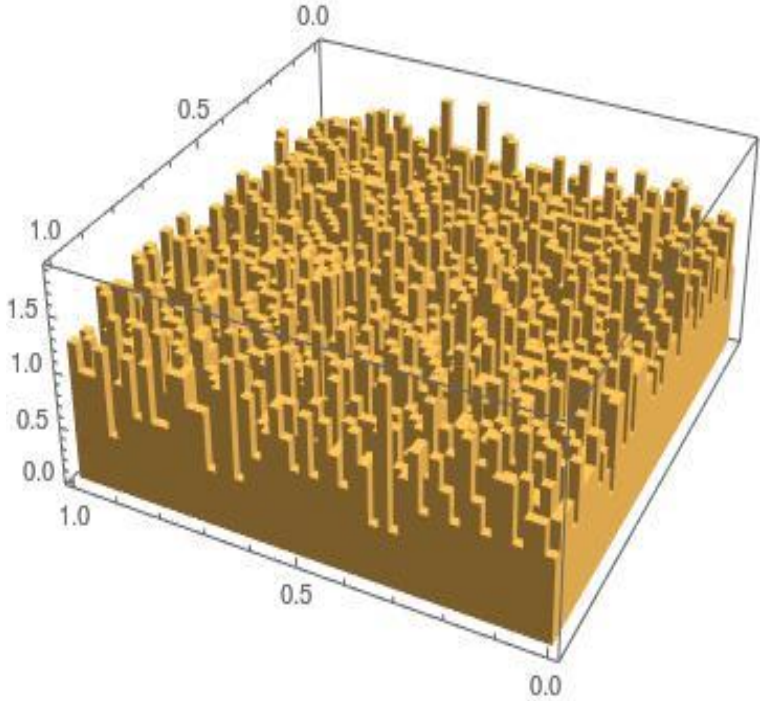
- Randqn1 (Numerical Recipies):
- Sobol (Wolfram Mathematica Documentation)
- MCG31(Rand() de distribuições antigas de C.

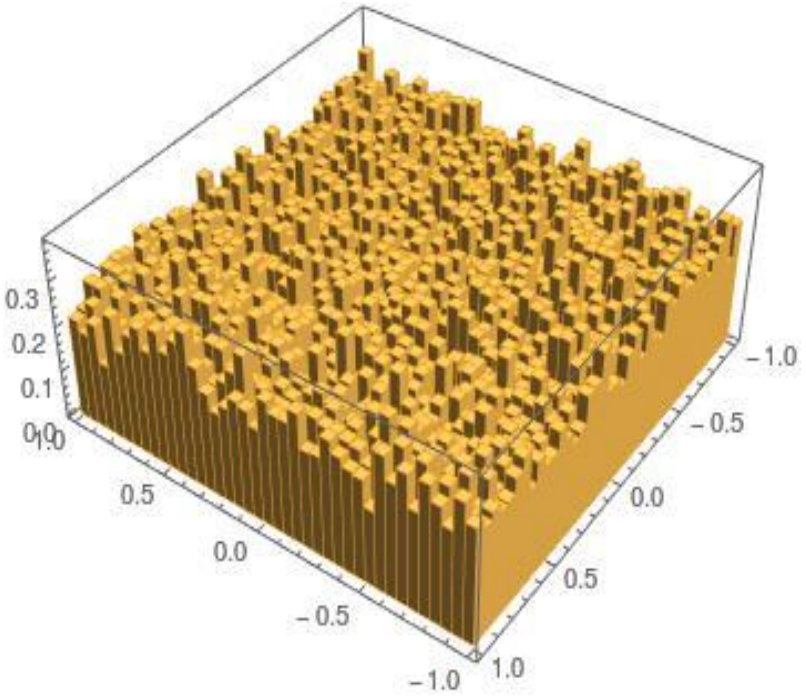
Sobol Vs Rand (6000 Pontos)

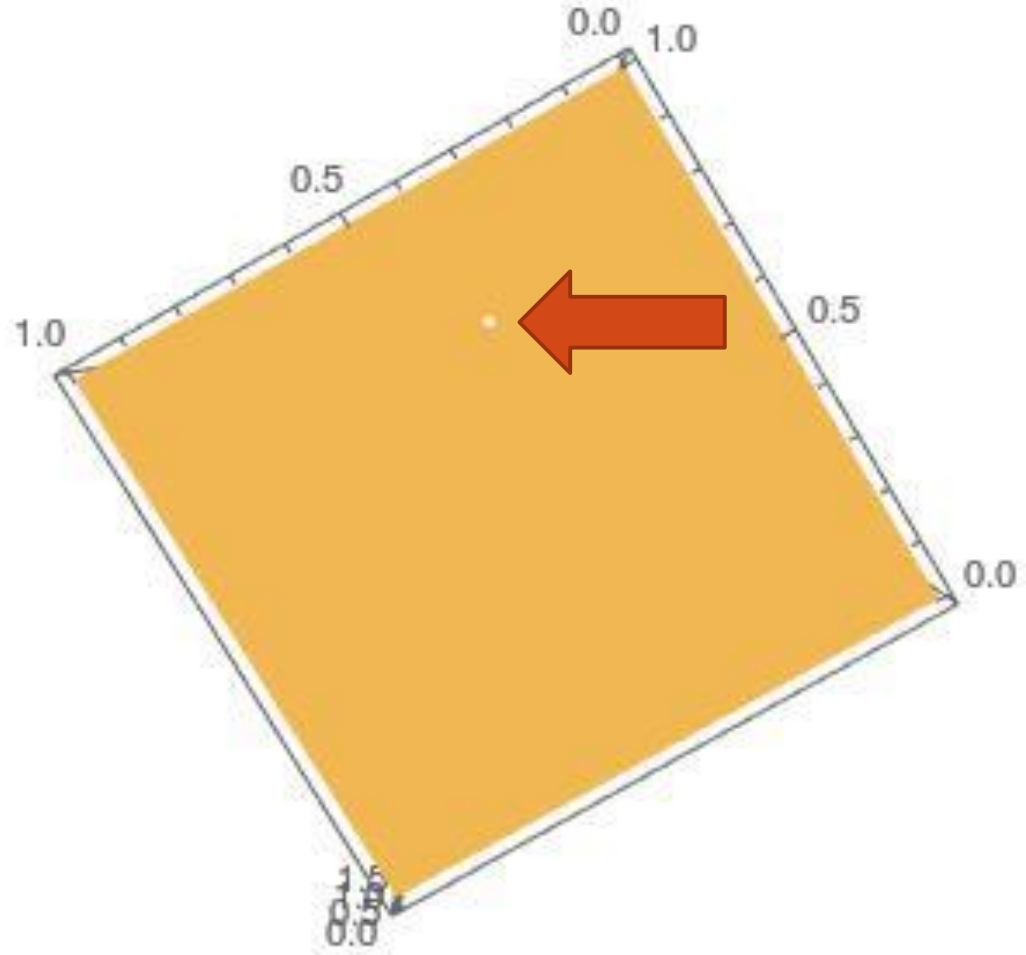




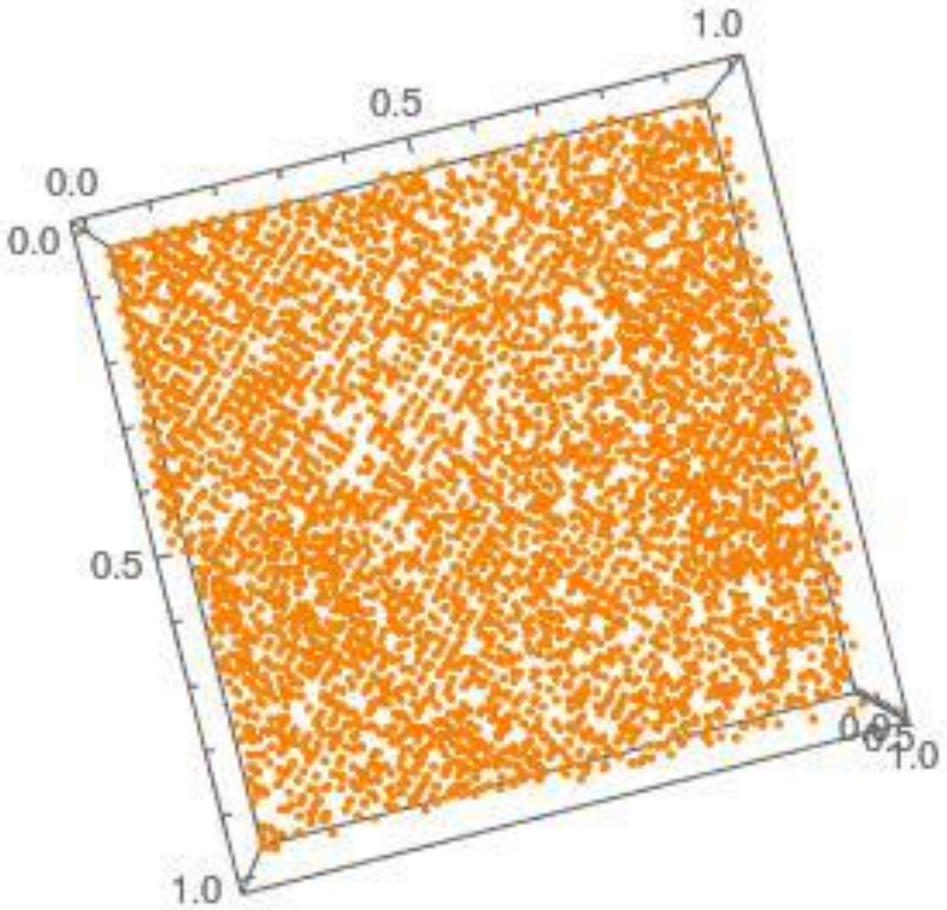
Sobol vs. Rand: Densidade de Pontos



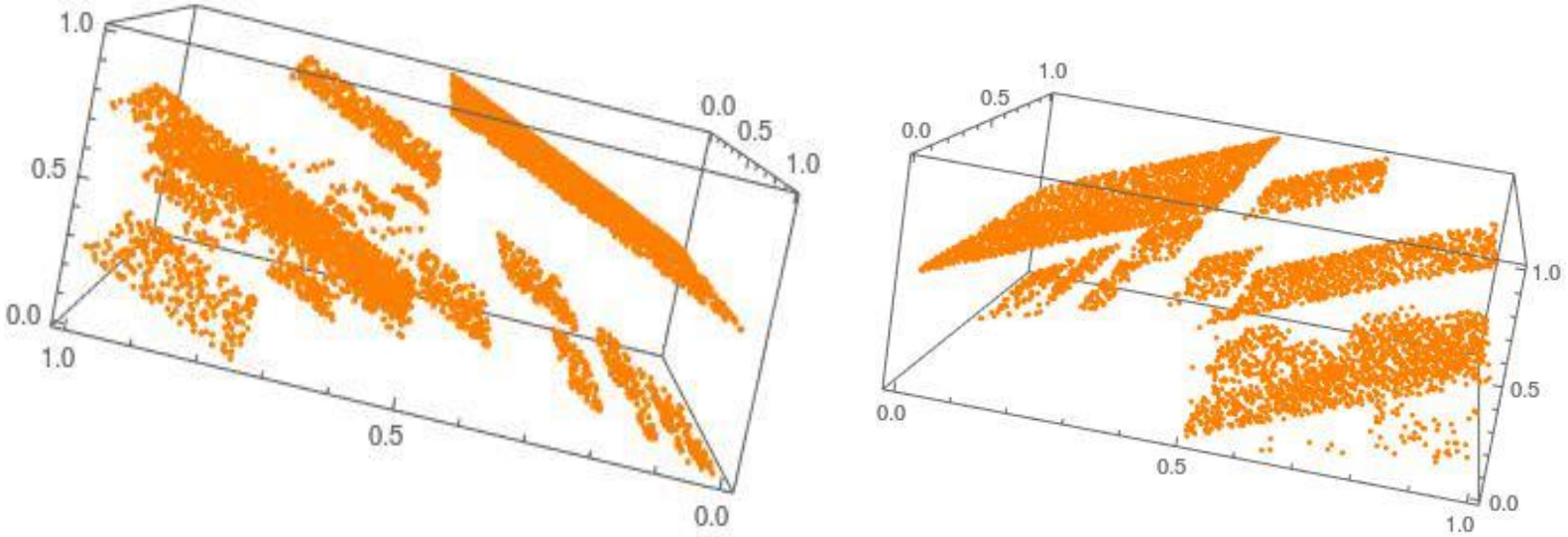



0.0
1.0
0.5
1.0
0.5
0.0

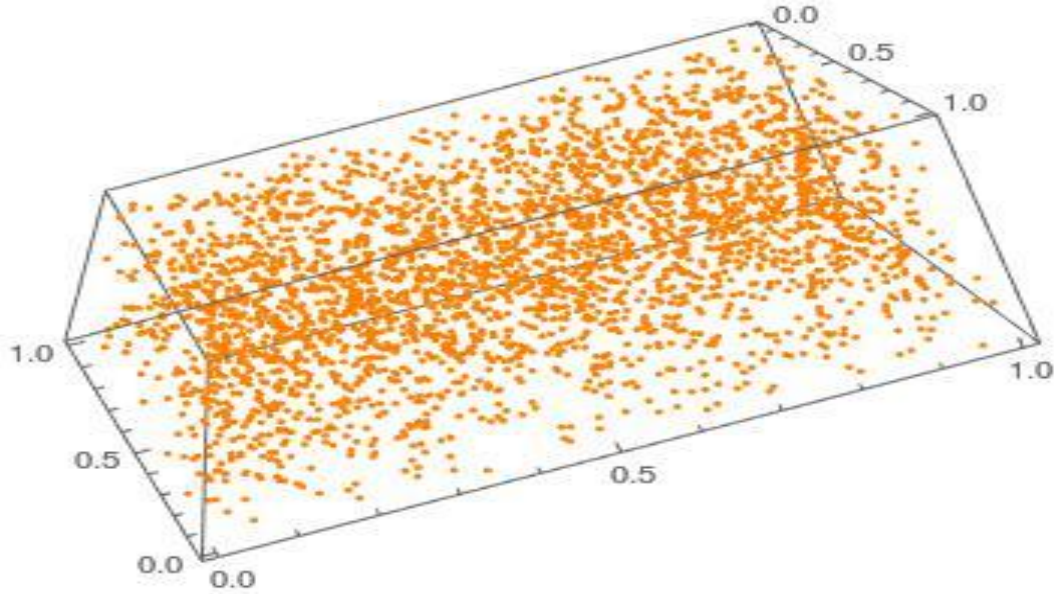
Sobol Plot em 3D



Sobol Plot em 3D



Sobol3D em 3D

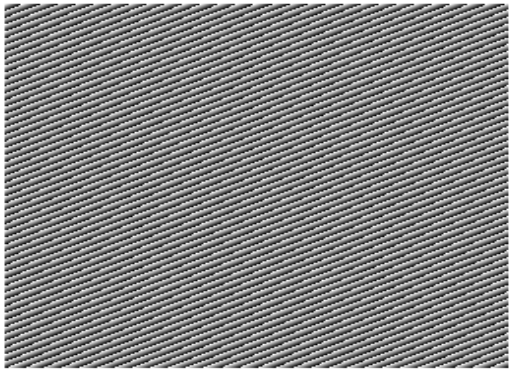


Sobol – Usos

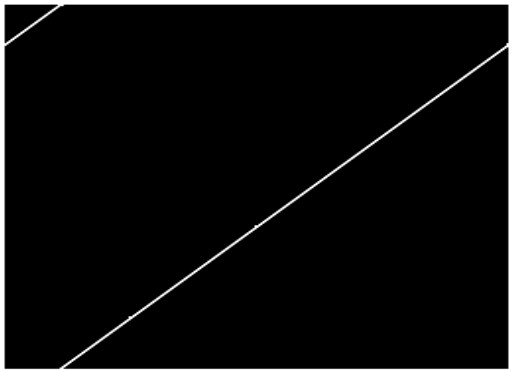
- Se olharmos para o espectro para poucos pontos do gerador Sobol vemos que ele é mais bem distribuído no espaço; para integrações numéricas isso é bom pois estamos dando importância igual para várias regiões. Geradores como esse são chamados de "*Low Discrepancy Generators*" e as simulações de *Quasi*-Monte Carlo.

Mais espectros

System.Random
100th number of seed 0...n

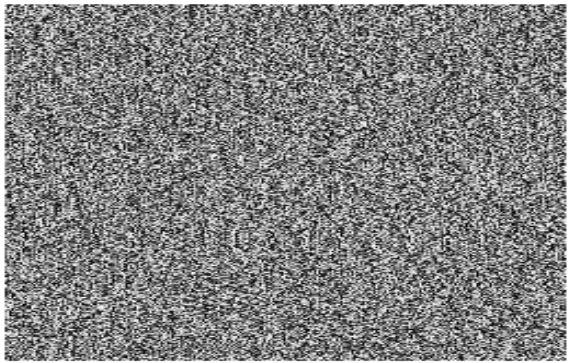


Sequence of 65536 random values.

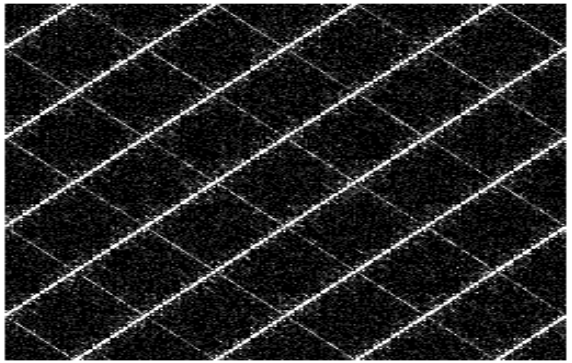


Plot of 500000 random coordinates.

PcgHash
numbers 0...n



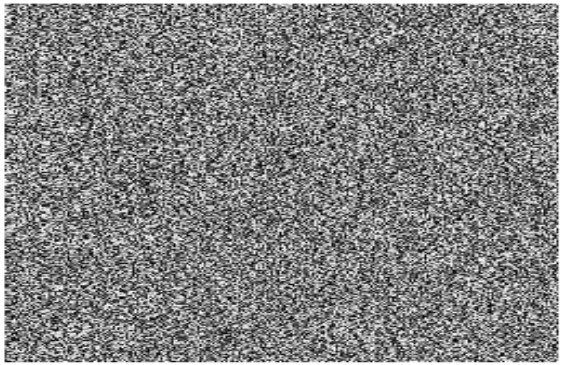
Sequence of 65536 random values.



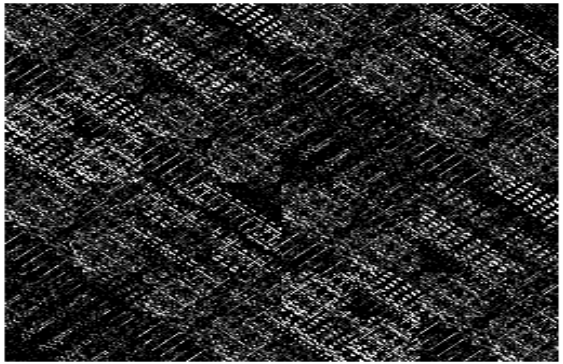
Plot of 500000 random coordinates.

Combinação de dois “ruins”

WangHash
numbers 0...n of seed 0

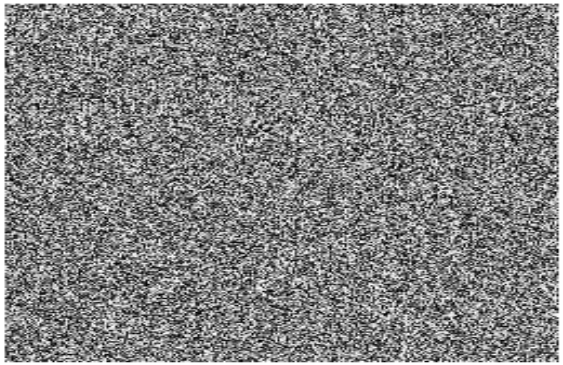


Sequence of 65536 random values.

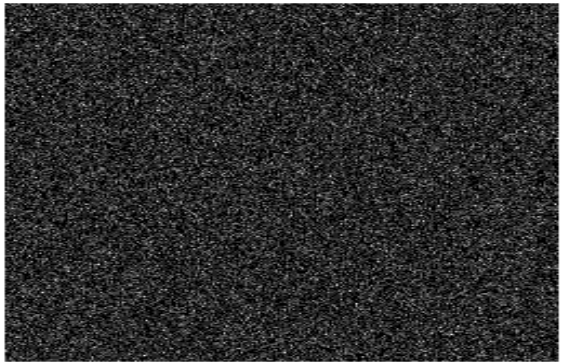


Plot of 500000 random coordinates.

WangDoubleHash
numbers 0...n of seed 0



Sequence of 65536 random values.



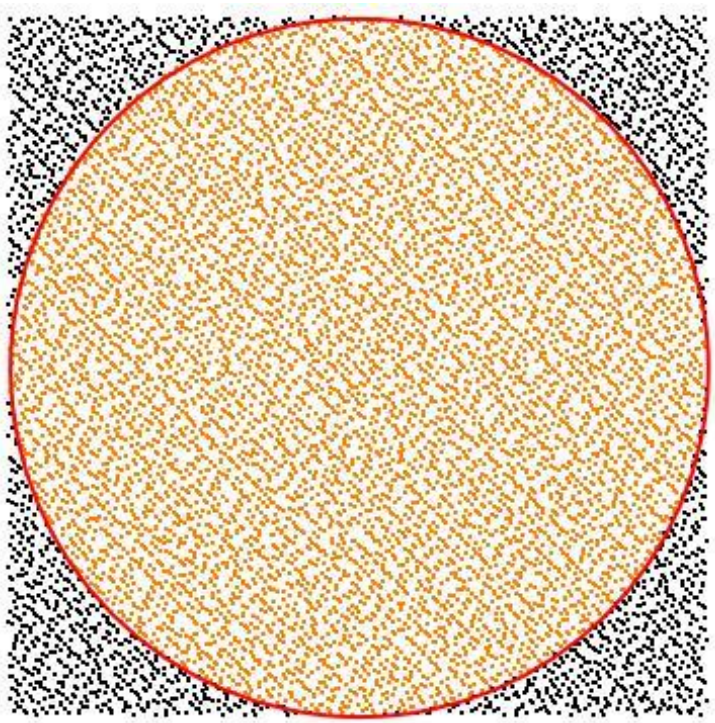
Plot of 500000 random coordinates.

Exemplos simples de testes

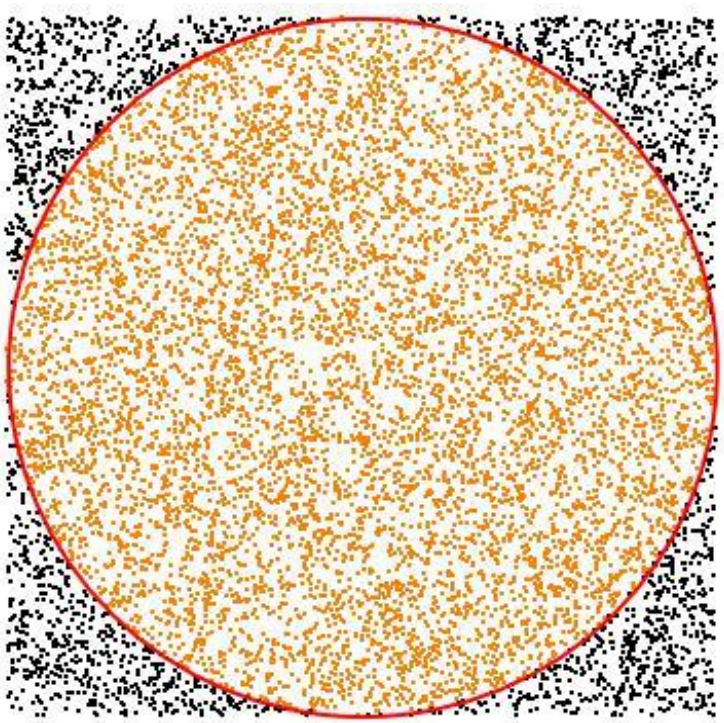
- Ache o valor de π plotando números em um quadrado e vendo quantos caem dentro de um circulo.

- Rand() (Implementação moderna em C) Vs. MCG31 (Multiplicative Congruential Gerarator 31, o Rand() antigo)

MCG31 vs Rand(): 10^5 pontos



$n = 10^5 \rightarrow \pi = 3.141(9)$
$n = 10^7 \rightarrow \pi = 3.1415(8)$



$n = 10^5 \rightarrow \pi = 3.141(9)$
$n = 10^7 \rightarrow \pi = 3.1415(5)$

Testes Mais conhecidos

- TestU01: Várias baterias de testes. Contem as baterias:

1. *Small Crush* (10 Testes $\approx$ 2 min)
2. *Crush* (96 Testes $\approx$ 2 Horas)
3. *Big Crush* (160 Testes $\approx$ 4 Horas)

- Diehard: 16 Testes

Testes mais adultos

- Vamos tomar analisar 4 exemplos da bateria *Diehard* e ver como os conceitos do curso de TEFE são utilizados nesse contexto.

Birthday Spacings

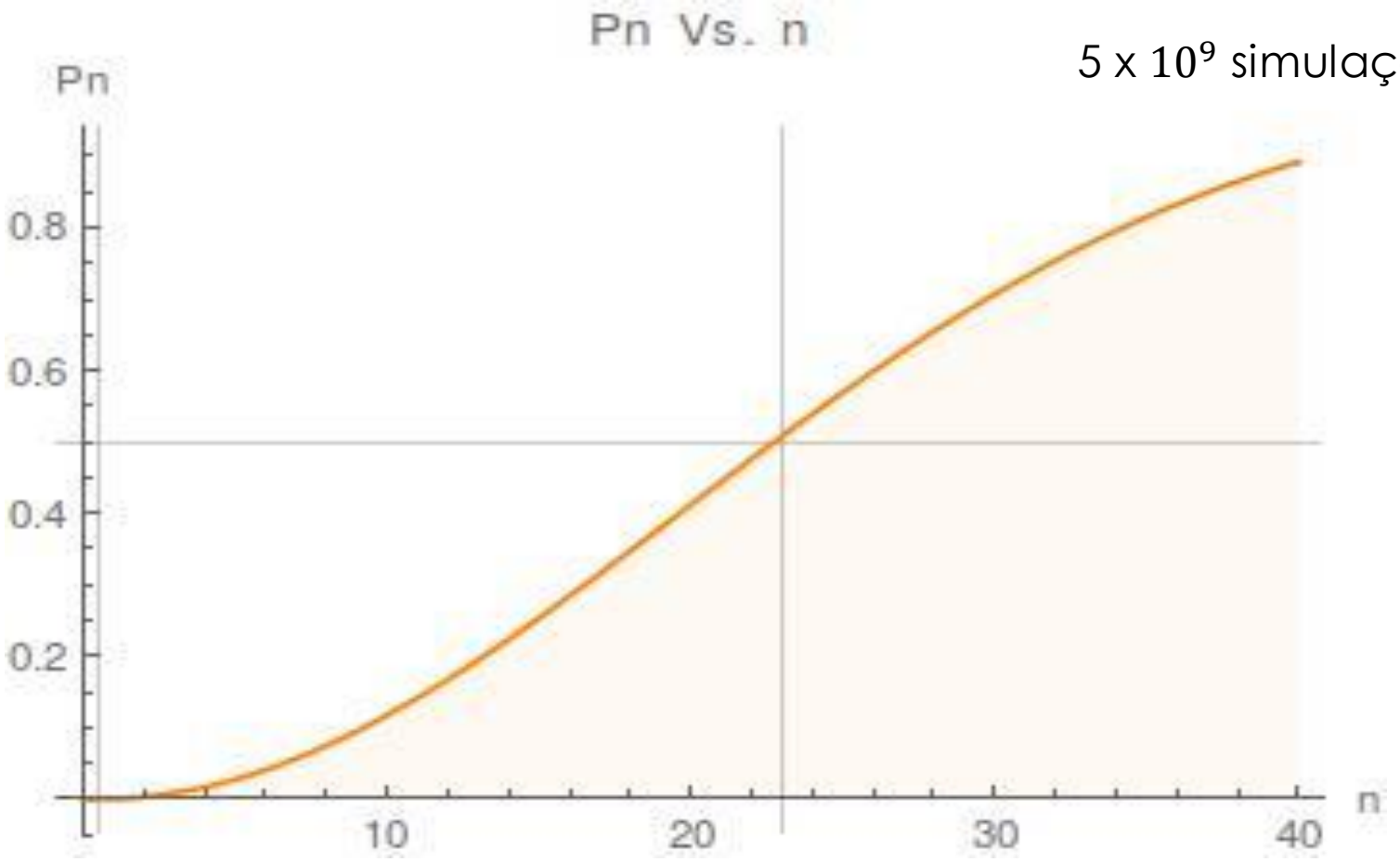
- "Qual a probabilidade de pelo menos duas pessoas de um grupo de n indivíduos escolhidos ao acaso tiverem o mesmo aniversário (dd/mm)?"

Teoria: Probabilidade pelo menos duas fazerem aniversário no mesmo dia:

- $p_1 = 1 - \frac{365}{365} = 0$
- $p_2 = 1 - \frac{364}{365} \approx 0.0027$
- $p_3 = 1 - \left(1 \times \frac{364}{365} \times \frac{363}{365}\right) \approx 0.0082$
- $p_n = 1 - \left(\frac{n!}{365^n} \binom{365}{n}\right)$

O teste computacional

- Escolha m aniversários em um ano de n dias.
- Liste as distâncias entre os aniversários.
- Seja j o número de aniversários coincidentes, então j segue assintóticamente a distribuição de Poisson (Paradigma de Poisson) de média $m^3/(4n)$.
- Documentação diz usar $n = 2^{24}$ e $m = 2^9$.
- 500 j's são feitos e usamos um teste χ^2 que nos fornece um valor p.
- Escolhemos grupos de 24 bits para os aniversários (1-24, 2-25, ...) ⇒ Vários valores de p para terminar o teste.



5 x 10^9 simulações

Infinite Monkey Theorem

- “Um macaco digitando aleatoriamente vai com probabilidades esmagadoras escrever qualquer texto possível”, inclusive a obra completa de Shakespeare, por exemplo.

Teoria

- Seja n o número de teclas, e a palavra desejada tenha m letras. A chance de digitar essa palavra é:
- $\left(\frac{1}{n}\right)^m$
- A chance de não digitar a palavra em j tentativas é:
- $p_j = \left(1 - \frac{1}{n^m}\right)^j$, que vai a 0 a medida que $j \to \infty$.

Teste computacional

- Gerar uma *stream* de 0 e 1, o número de 1's em um byte será uma "letra". Em 10^7 letras, contar a frequência de cada "palavra" de 5 letras. Analisar a correlação.

"This is a thousand monkeys working on a thousand typewriters. Soon, they'll have writen the greatest novel know to man."

"It was the best of times, it was the "blurst" of times?! You stupid monkey!"

Outros testes (Cultura Geral)

- Ranks of matrices: Geramos matrizes (31 x 31, 32 x 32, 6 x 8) e analisamos o *rank* delas (dimensão do espaço gerado pelas colunas). Um teste de chi quadrado é realizado sobre o valor do *rank* de várias matrizes (ranks menores que 28 são muito raros).
- Parkinglot. "Estacionar" círculos em um quadrado. Plotar número de sucessos contra tentativas. PDF é desconhecida porém a soma deve seguir uma Gaussiana.
- Minimum distance test: Plotar pontos em um quadrado. O quadrado da distância mínima entre os $(n^2 - 1)/2$ pares deve seguir um distribuição exponencial se os dados forem homogeneamente distribuídos
- Overlapping sums test: gerar *floats* em $(0,1)$ e somar de 100 em 100. A somas devem seguir uma Gaussiana de média e desvio conhecidos

Resultados numéricos

- *Long story short*:
- Geradores listados como “ruins”: “““falham”””
- Geradores listados como bons/modernos passam.

Motivação: Geradores de números aleatórios por hardware

- Também chamados de Geradores de Números Verdadeiramente Aleatórios (TRNG na sigla em inglês)
- Exploram fenômenos físicos aleatórios
- Uso mais comum na área de encriptação de dados, onde são utilizados para gerar chaves criptográficas (urna eletrônica)

Proposta: Investigar propostas de geradores de numeros aleatorios utilizando o Arduino

- Implementação da função random() diretamente da biblioteca avr-libc
- Determinismo: retorna a mesma lista a cada inicialização do Arduino

Contra-proposta: Investigar alternativas de alimentação da randomSeed() do Arduino

- Ambas as alternativas são tentativas diferentes de tirar proveito de um mesmo fenômeno físico: flutuações aleatórias nos valores de tensão elétrica medidos pela função analogRead.

Contra-proposta: Investigar alternativas de alimentação da randomSeed() do Arduino

1 - Referência: “If it is important for a sequence of values generated by random() to differ, on subsequent executions of a sketch, use randomSeed() to initialize the random number generator with a fairly random input, such as analogRead() on an unconnected pin.”

Contra-proposta: Investigar alternativas de alimentação da randomSeed() do Arduino

2 - Comunidade: Compor números utilizando o bit menos significativo (LSB na sigla em inglês) do valor retornado pela analogRead() quando aplicada a um pino desconectado

analogRead(): Funcionamento

- Todos os 6 pinos analógicos conectados ao mesmo ADC (conversor de sinais analógicos para digitais)
- ADC realiza uma busca binária de 10 bits, resultando em 1024 canais.
- Cada canal é associado a uma tensão por meio de C*T_ref/1024, onde C é o canal retornado pelo ADC, T_ref a tensão de referência

analogRead(): Funcionamento

- Tensão de referência: no caso da alimentação via USB, é utilizada a tensão desta. Quando a alimentação provém de outra fonte, é fornecida ao ADC pelo regulador de tensão presente no circuito (maior precisão).
- Regulador de tensão apresenta melhor funcionamento para tensão de entrada superior a 7 volts.

Planejamento

- Isolamento da rede elétrica: Para cada alternativa foram coletadas duas séries de dados, uma com o sistema alimentado por uma bateria de 9v, outra pela USB de um computador. Para eliminar a necessidade de comunicação serial os dados foram armazenados em cartão micro-SD.
- Distanciamento de potenciais fontes de erro dominantes, como emissores de rádio (celulares, roteadores), motores elétricos (ventiladores), etc.

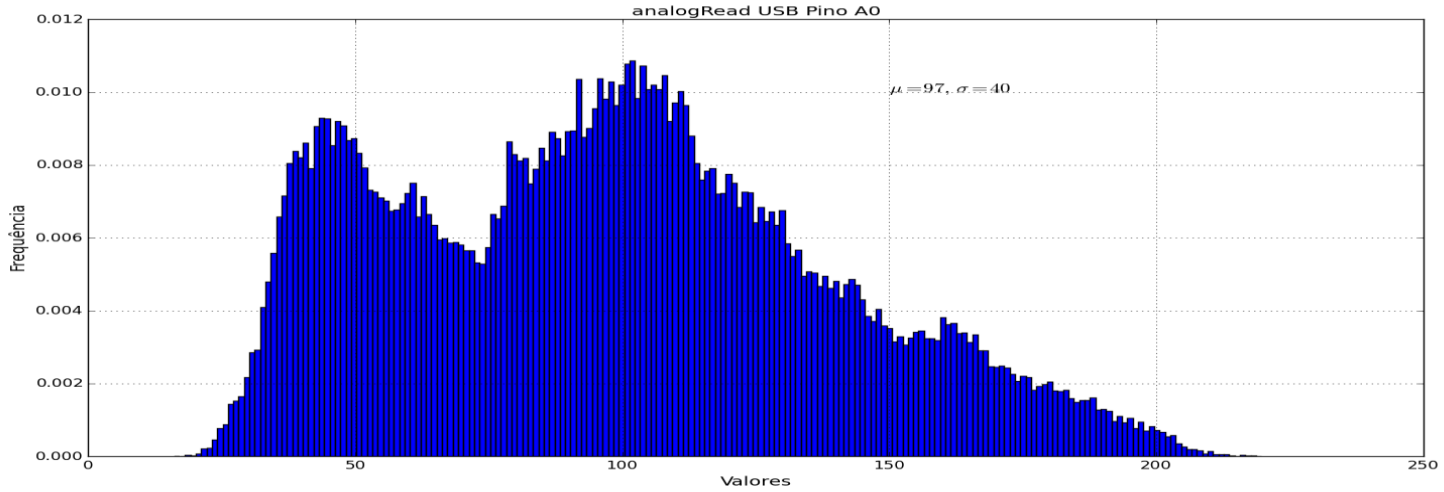
Planejamento

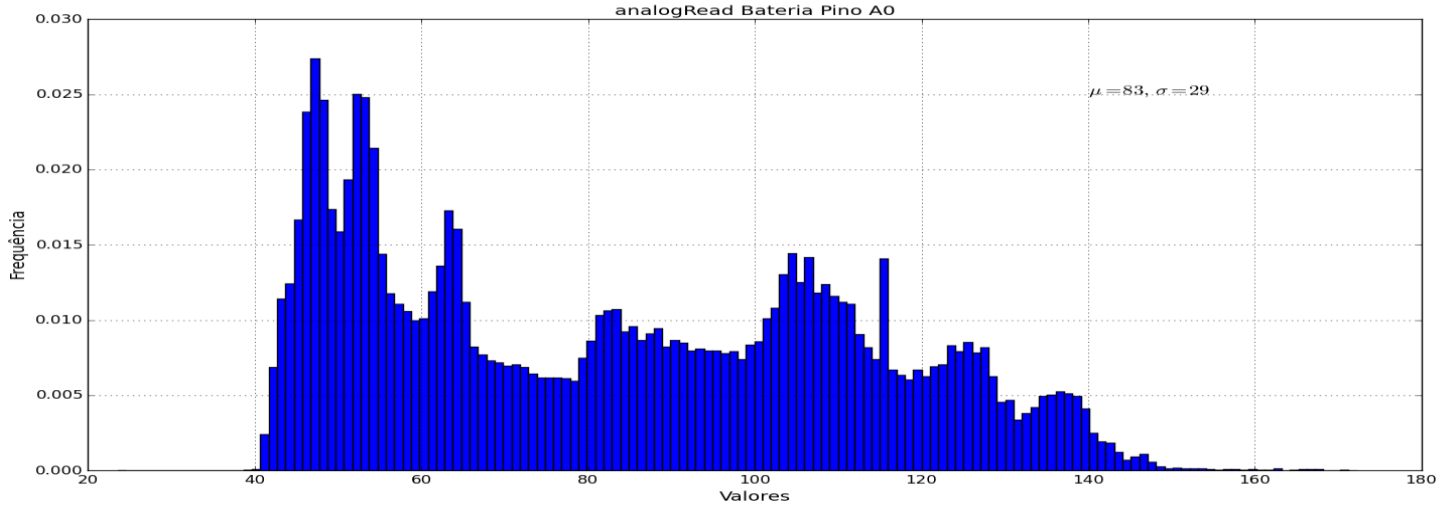
- Medidas realizadas em todos os 6 pinos analógicos (A0 à A5), para tentar identificar eventuais fontes de erro dominantes com origem no próprio circuito

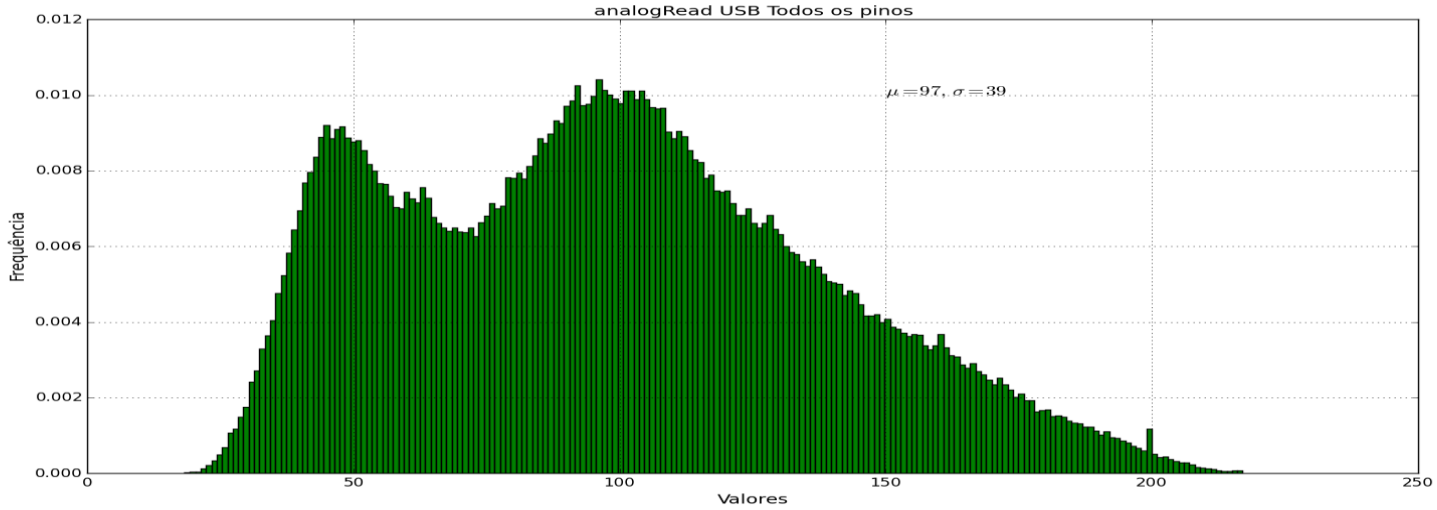
- Determinação do tempo necessário para realização de mil medidas por pino, para posterior estimativa do número de coletas que utilizasse 10 minutos “de máquina”

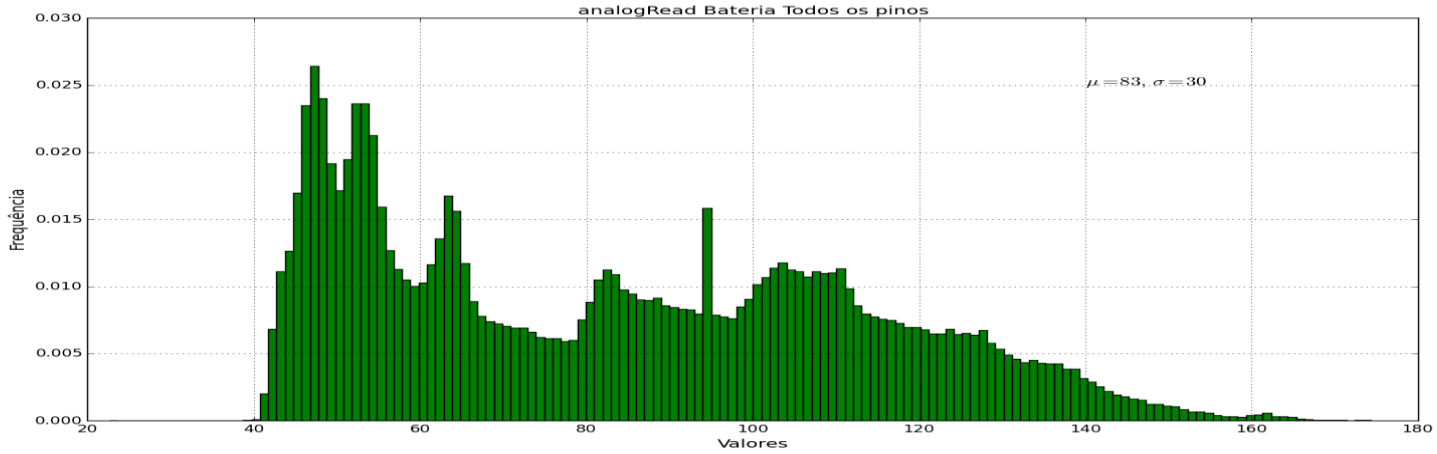
Dados obtidos para a alternativa 1 (Referência)

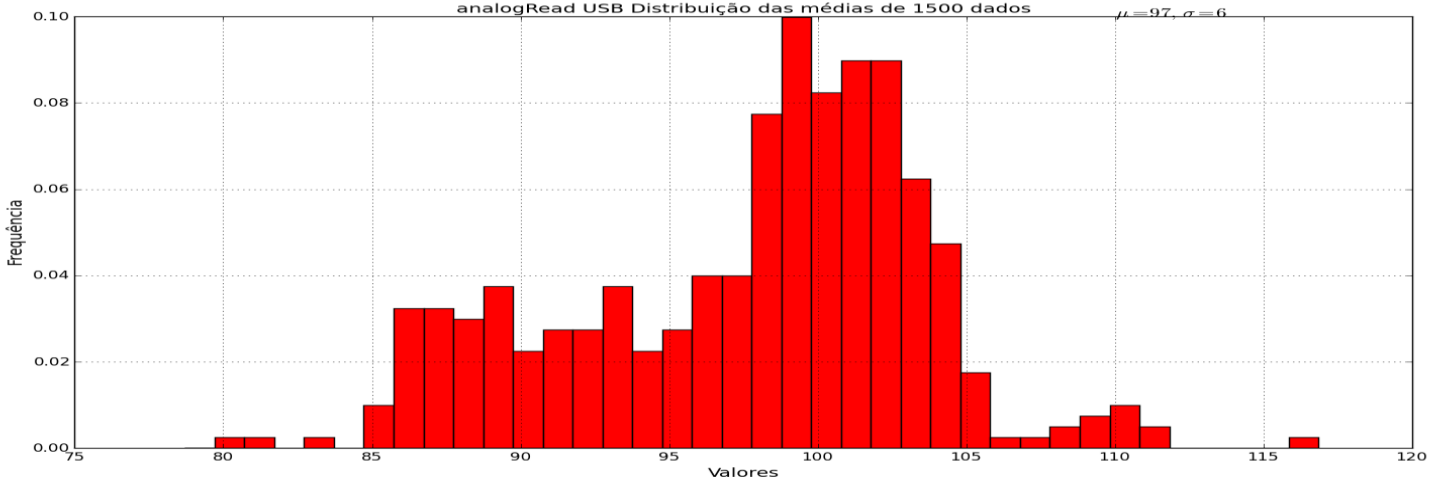
- 100 mil dados por pino, 600 mil dados no total
- Por volta de 6 segundos por mil medidas

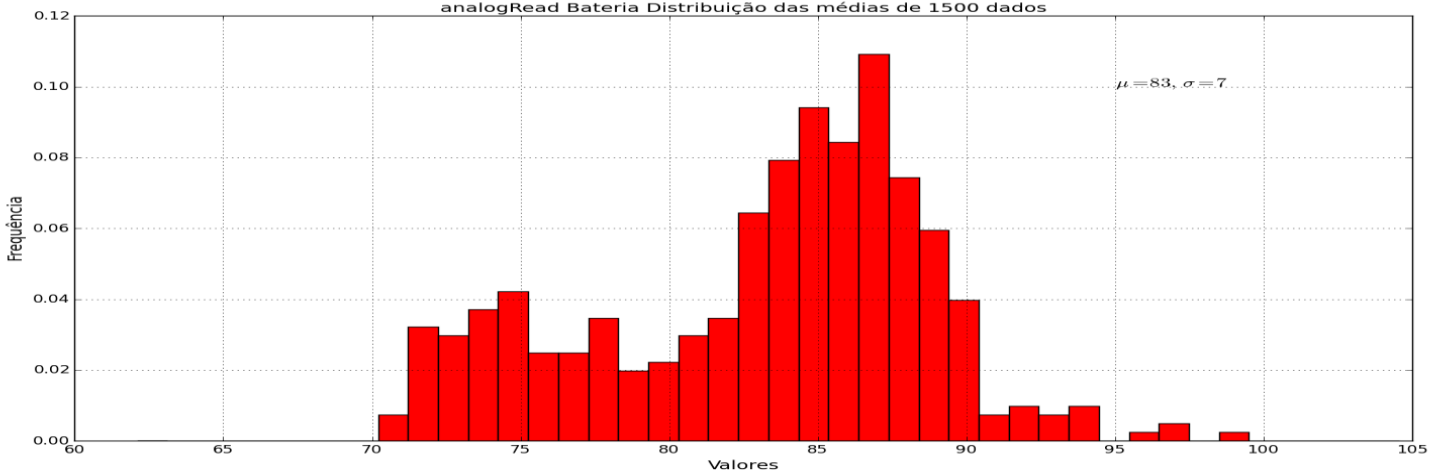

analogRead USB Pino A0
$\mu=97,\ \sigma=40$
Frequência
Valores
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250


analogRead Bateria Pino A0
$\mu=83,\ \sigma=29$
Frequência
Valores
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
120
140
160
180


analogRead USB Todos os pinos
$\mu=97, \sigma=39$
Frequência
Valores
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250


analogRead Bateria Todos os pinos
$\mu=83, \sigma=30$
Frequência
Valores
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
120
140
160
180


analogRead USB Distribuição das médias de 1500 dados
$\mu=97, \sigma=6$
Frequência
Valores
0.00
0.02
0.04
0.06
0.08
0.10
75
80
85
90
95
100
105
110
115
120

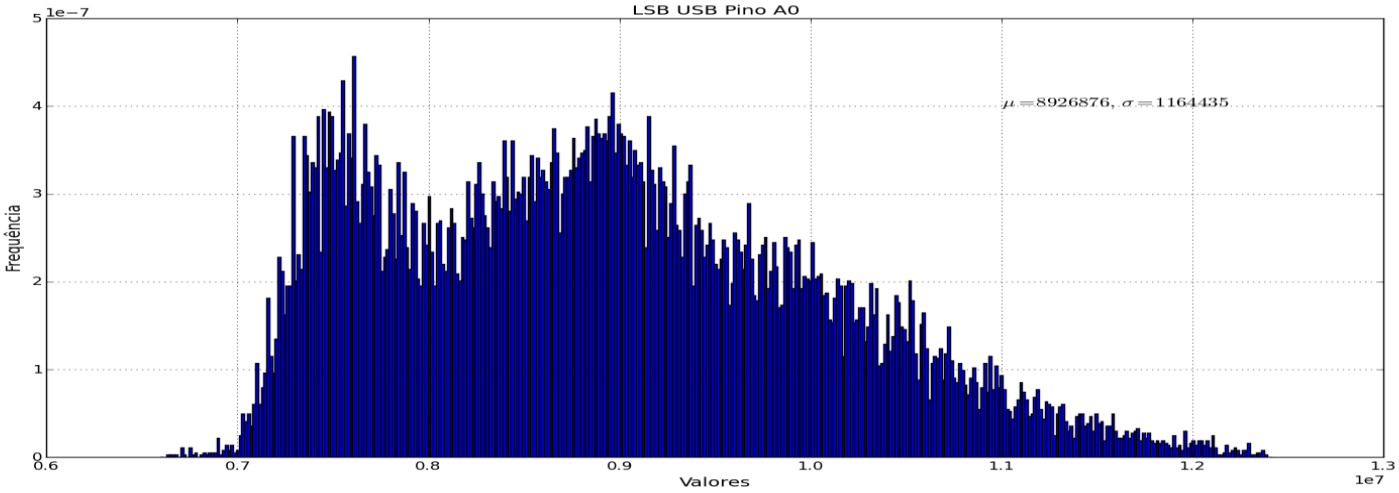

analogRead Bateria Distribuição das médias de 1500 dados
$\mu=83, \sigma=7$
Frequência
Valores
0.00
0.02
0.04
0.06
0.08
0.10
0.12
60
65
70
75
80
85
90
95
100
105

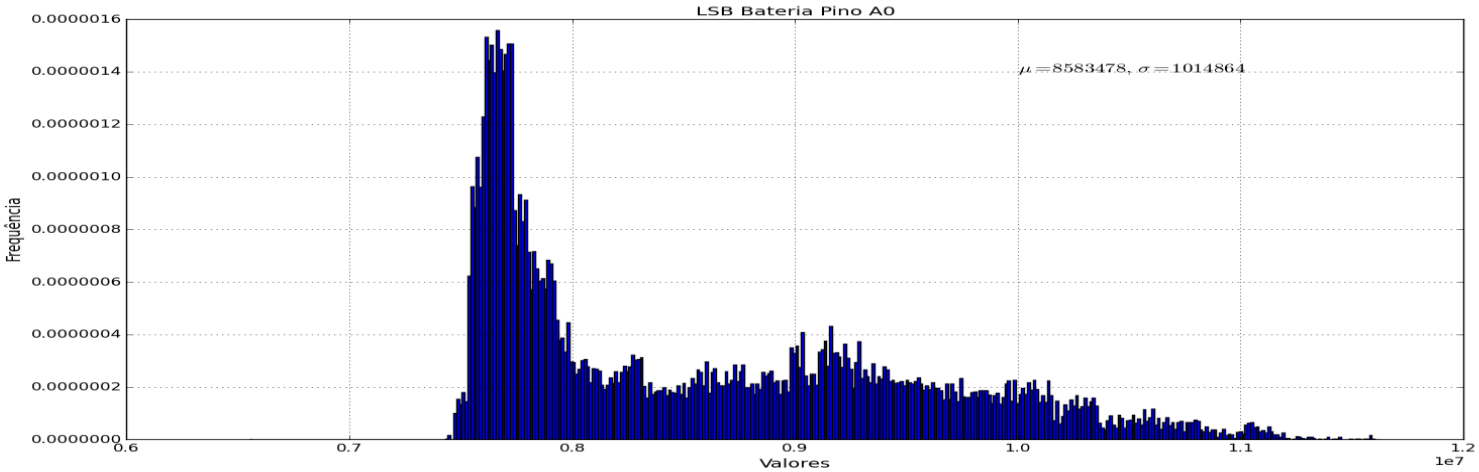
Análise

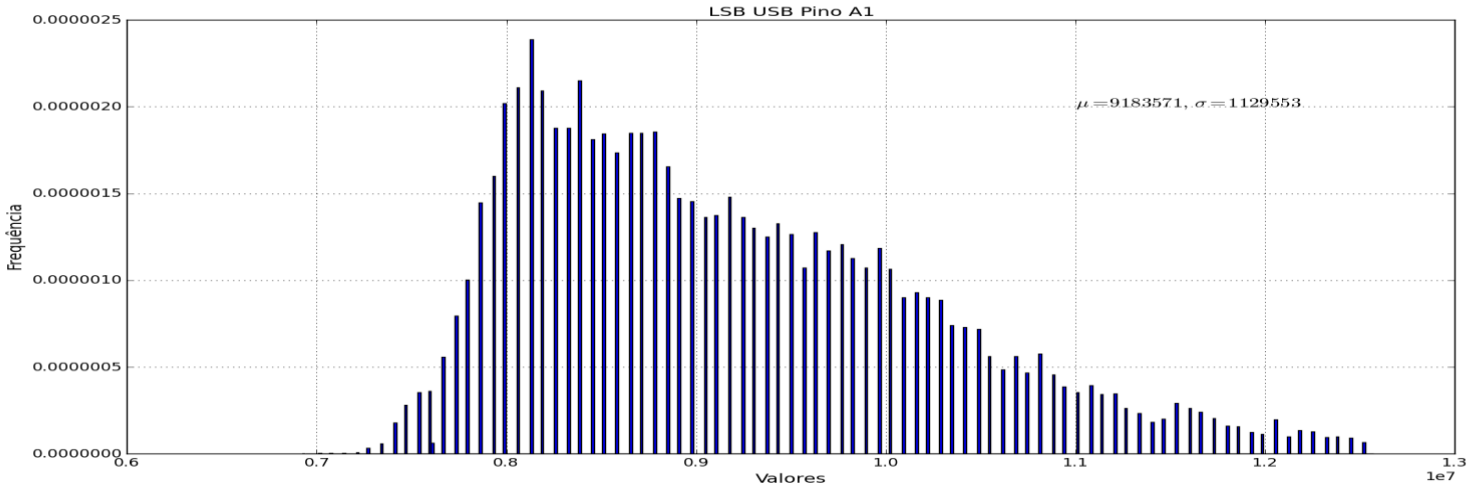
- Diferença notável nas distribuições dos dados originais
- Eliminação da fonte de erro associada a alimentação via USB implicou em uma distribuição menos suave
- Semelhança notável entre as distribuições das médias, com sigmas semelhantes e médias deslocadas
- Bimodalidade: Disjunção lógica de duas variáveis aleatórias = Dados oscilam ao redor de dois valores mais prováveis de tensão

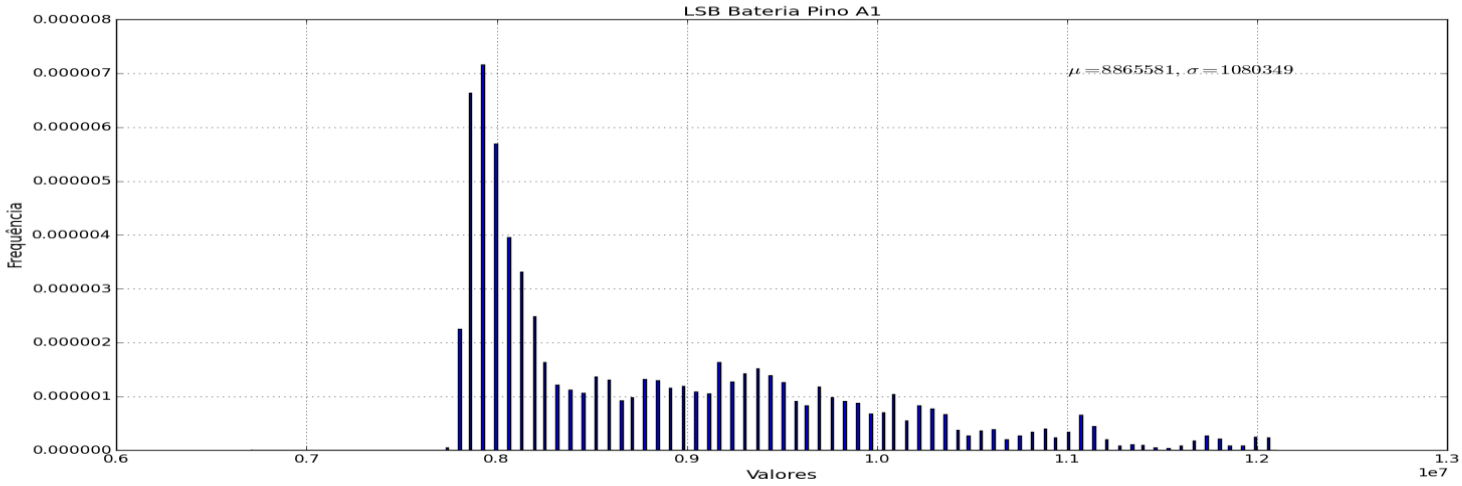
Dados obtidos para a alternativa 2 (Comunidade)

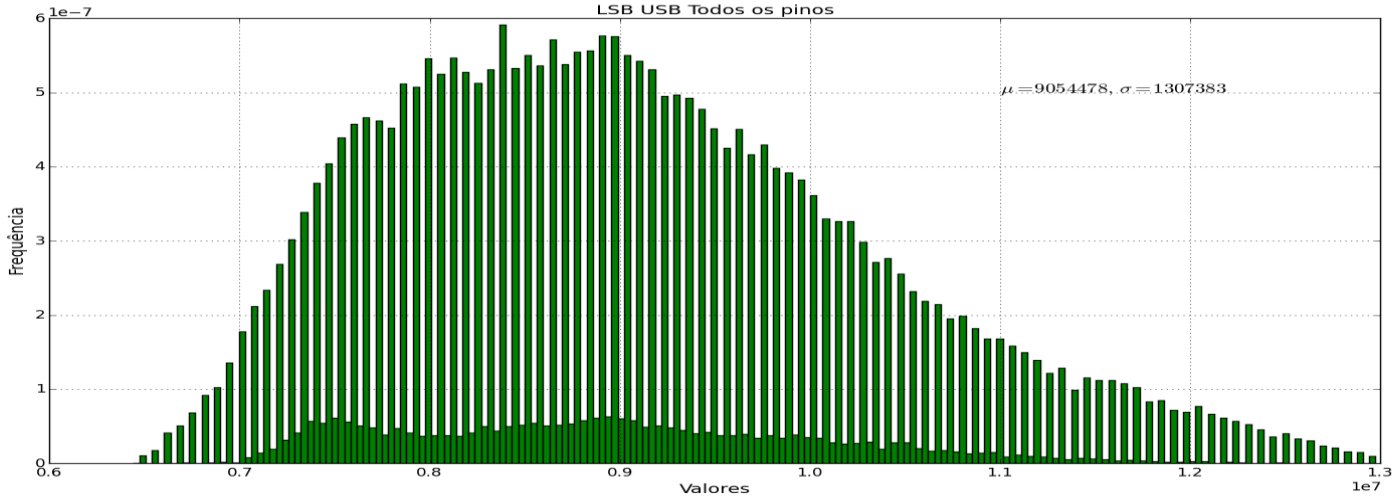
- 25 mil dados por pino, total de 150 mil dados
- Por volta de 24 segundos por mil medidas
- Cada número gerado corresponde a concatenação dos bits menos significativos de 32 valores retornados pela analogRead()

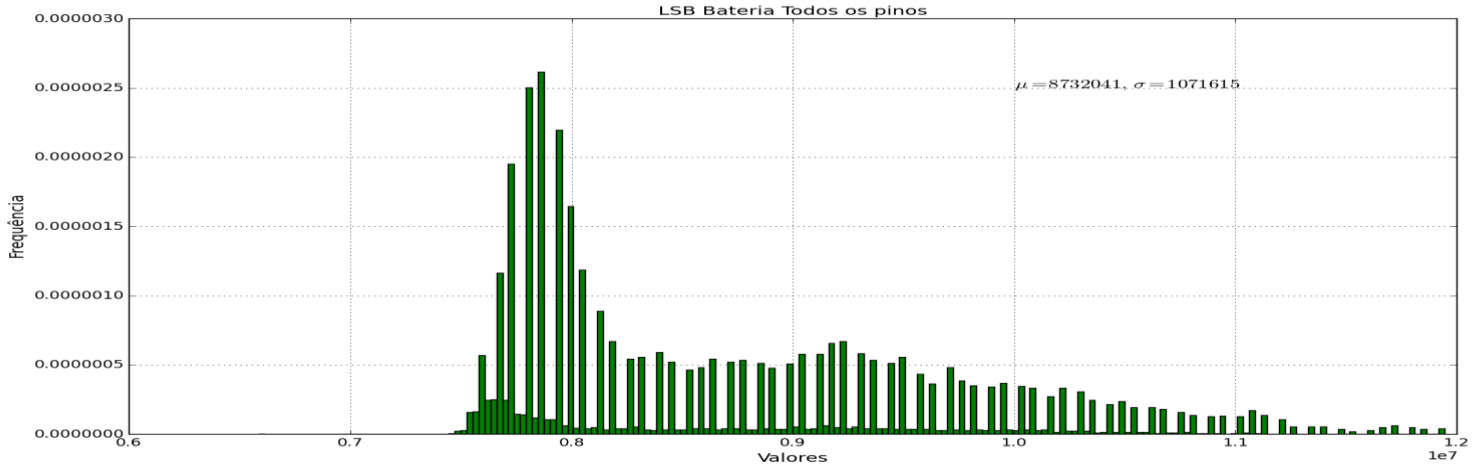

LSB USB Pino A0
1e−7
$\mu = 8926876$, $\sigma = 1164435$
Frequência
Valores
1e7

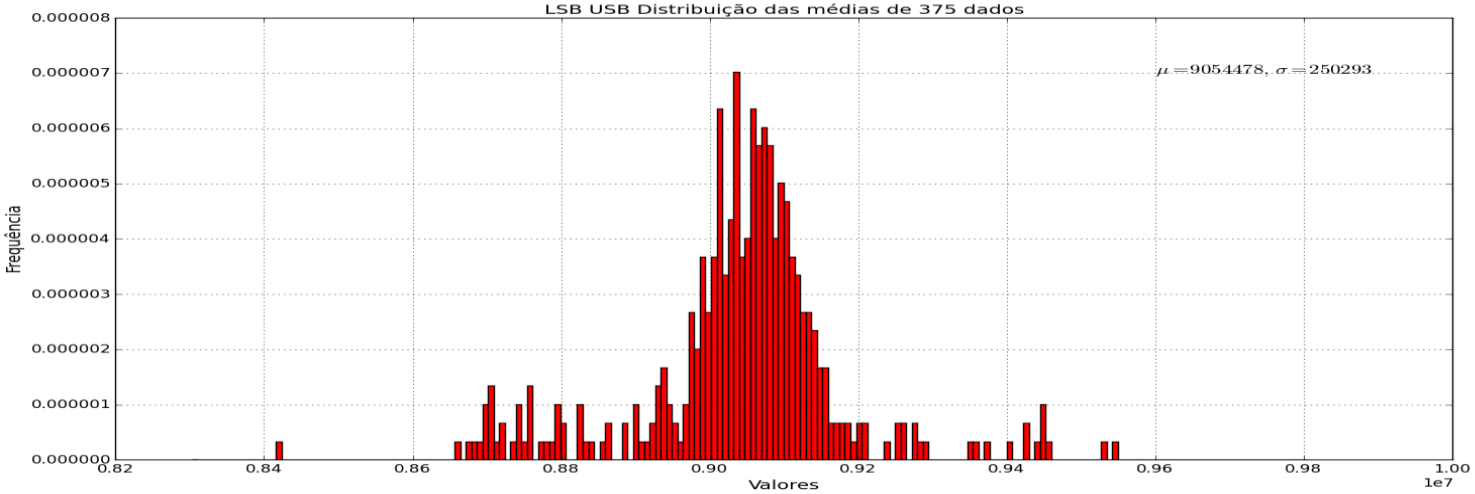

LSB Bateria Pino A0
$\mu = 8583478$, $\sigma = 1014864$
Frequência
Valores
1e7

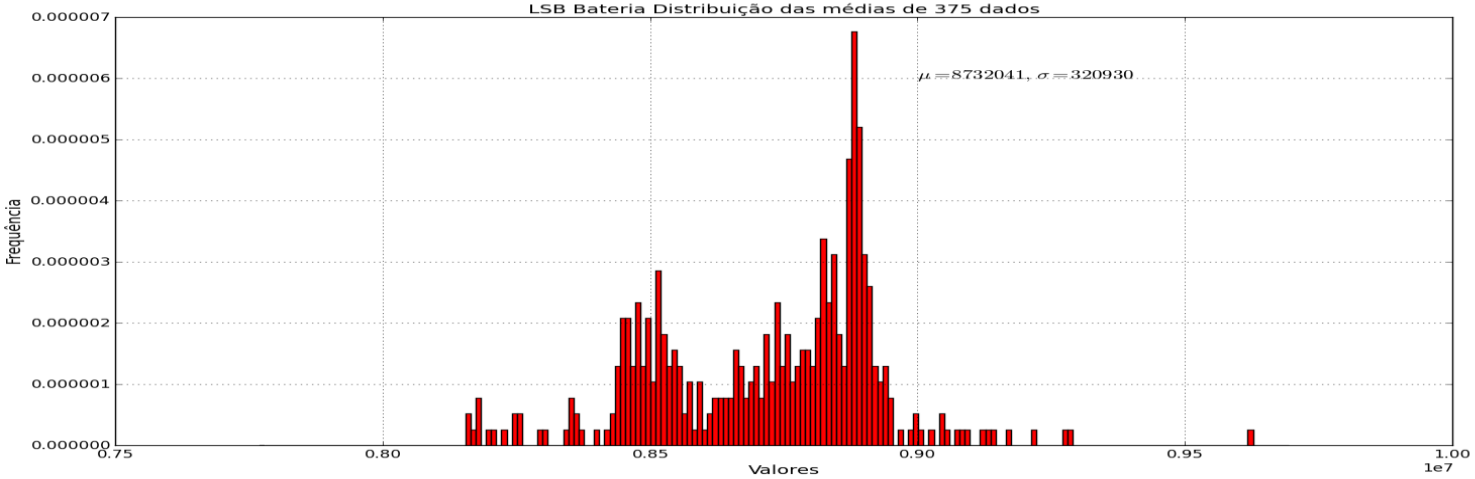

LSB USB Pino A1
μ=9183571, σ=1129553
Frequência
Valores
1e7


LSB Bateria Pino A1
μ=8865581, σ=1080349
Frequência
Valores
1e7


LSB USB Todos os pinos
1e−7
μ=9054478, σ=1307383
Frequência
Valores
1e7


LSB Bateria Todos os pinos
μ=8732041, σ=1071615
Frequência
Valores
1e7


LSB USB Distribuição das médias de 375 dados
$\mu = 9054478$, $\sigma = 250293$
Frequência
Valores
1e7


LSB Bateria Distribuição das médias de 375 dados
$\mu = 8732041$, $\sigma = 320930$
Frequência
Valores
1e7

Análise

- Distribuições para os pinos diferentes de A0 apresentam intervalos regulares onde não ocorre nenhuma observação! Hipótese: não ocorrência de alguma sequência binária durante a construção bit a bit dos valores.
- Pino A0 não apresenta os mesmos intervalos proibidos! Hipótese: presença ou ausência de fonte de ruído, permitindo que a(s) sequência(s) binárias ausentes nos demais pinos ocorram.
- Impacto da localização do pino na distribuição, mesmo as medidas sendo realizadas pelo mesmo ADC.

Análise

- Distribuições dos dados originais notavelmente distintas para as diferentes fontes de alimentação.
- Distribuição das médias com alimentação via bateria apresenta bimodalidade invertida em relação aos dados! Hipótese: o valor associado ao pico suave observado à direita nas distribuições dos dados possui variância significativamente maior que o pico mais notável, porém os dados flutuavam ao redor da média associada a este pico com maior frequência.
- Distribuição (aparentemente) monomodal para alimentação via USB, com assimetria negativa. Hipótese: Impacto significativo da fonte de alimentação na oscilação do bit menos significativo

Conclusões

- Ambos os métodos fornecem sementes ao gerador de números aleatórios que possuem uma distribuição de probabilidade, podendo ser consideradas variáveis aleatórias.
- Os espaços amostrais das duas propostas mostraram-se significativamente pequenos, o que fica evidenciado pela comparação do desvio padrão com com 2^N_bits

Conclusões

- Dependendo do número de sementes geradas no contexto de um eventual projeto, as alternativas podem ser razoavelmente satisfatórias.
- A construção dos números pelo bit menos significativo evidenciou o fato, comentado no decorrer do curso, que a sobreposição de fontes de erro aleatórias aproxima a distribuição da soma desses a uma distribuição normal.

Referências

- http://www.sunshine2k.de/articles/Birthday_Paradoxon.pdf
- http://fibonacci-seri.es/post/3170083113/fibonacci-sequence-binary-plot
- https://www.doc.ic.ac.uk/project/examples/2012/163/g1216326/applications.php
- https://en.wikipedia.org/wiki/Rule_30
- Patrick Lacharme *et al*. "Pseudo-random sequences, boolean functions and cellular automata. J-F. Michon, P. Valarcher, J-B. Yun`es. Boolean Functions: Cryptography & Applications", Presses Univ ersitaires de l'Univ ersit´e de Rouen, pp.80-95, 2008.
- http://reference.wolfram.com/language/tutorial/RandomNumberGeneration
- https://en.wikipedia.org/wiki/Mersenne_Twister
- Arduino Reference: random()
- Arduino Reference: analogRead()
- Arduino Forum
- AVR Libc
- Formula / calculation of the function random() / randomSeed()
- (In)segurança do v oto eletrônico no Brasil / Vulnerabilidades no software da urna eletrônica brasileira