A Fast, Cheap, High-Entropy Source for IoT Devices

Ben Lampert, Riad Wahby, Shane Leonard,Phil Levis

Introduction - How do you evaluate random number generators (RNG)

Entropy is a measure of an adversarial information on a sequence of bits given knowledge of how your random bits are being generated. Few important measures of random bit streams:

-Bias and Shannon entropy (Probability distribution)

-Serial Correlation

-1bit of entropy per bit is ideal

Thoughts on random number generation

"Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin."
-John Von Neumann (Mathematician)

"Relying solely on the hardware random number generator which is using an implementation sealed inside a chip which is impossible to audit is a BAD idea."

-Theodore Tso (Kernel Developer)

Why build our own?

-Entropy pools in modern OS’s have a lot of entropy sources to draw from (Hard drive timing, user inputs, incoming packet timing, etc).

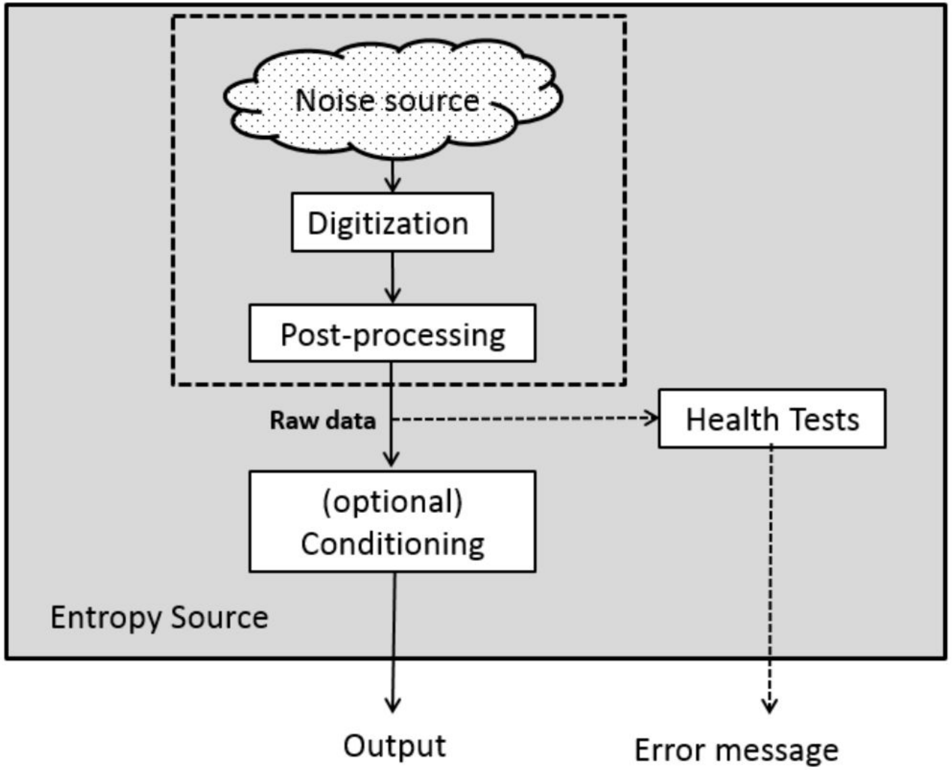
-Embedded IoT devices have less ways to gather entropy, therefore those sources of entropy must be very good.

-IoT devices have unique power and size constraints

-Internal rand() type instructions can obfuscate where the entropy is coming from, so for security applications would be nice to make this transparent.

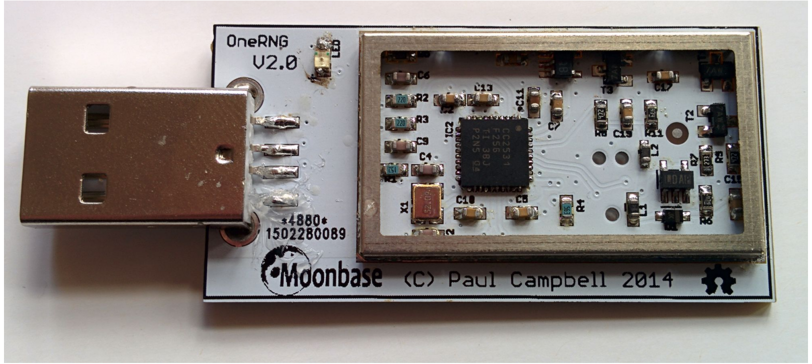
The HWRNG Approach

1) Take a noise source (Thermal noise, radiation, radio noise, semiconductor noise)
2) Amplify noise source (if necessary)
3) Digitize the noise source
4) Check health
5) Debias/Condition



http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf

Existing HWRNG Devices (OneRNG)

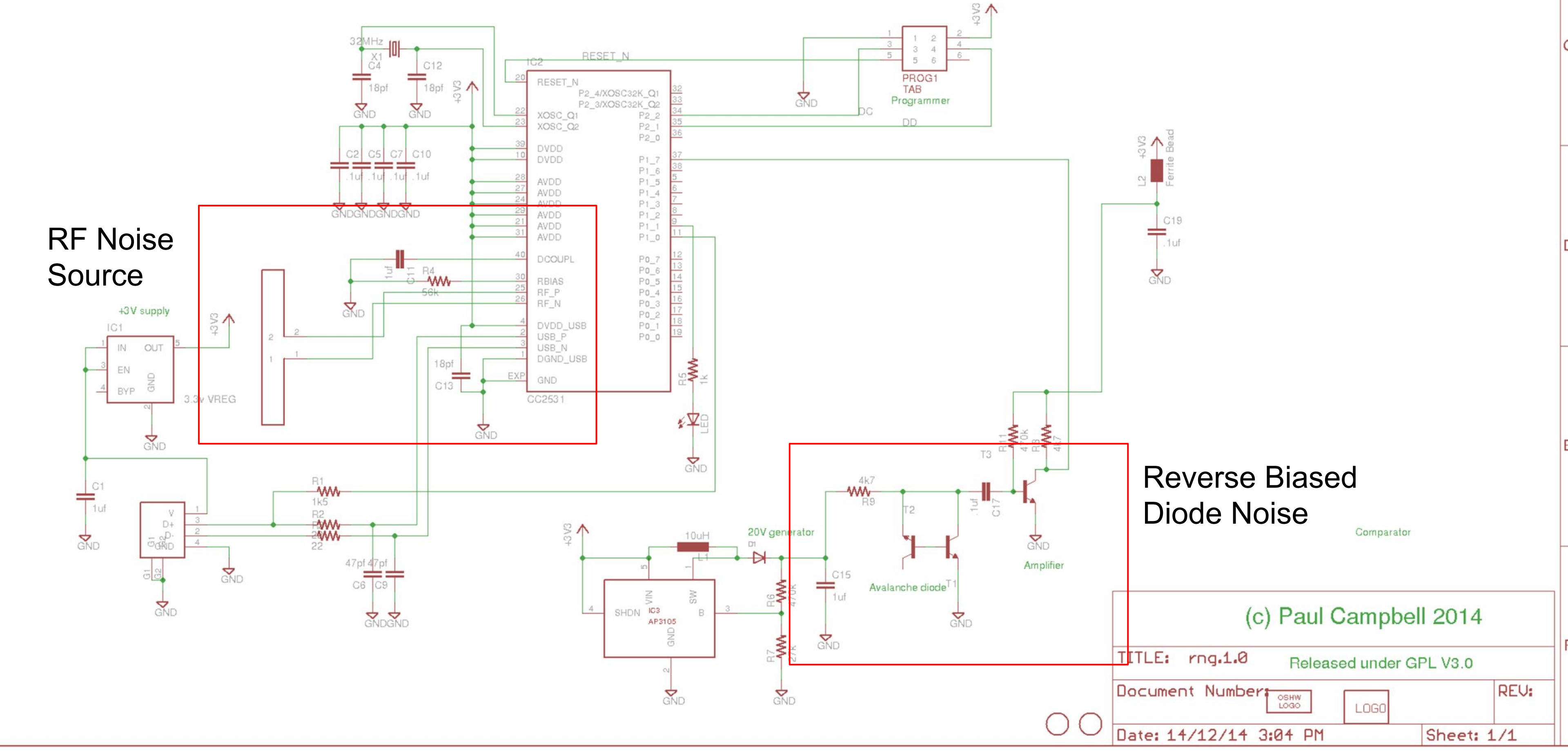


OneRNG (http://onerng.info/)

OpenSource design

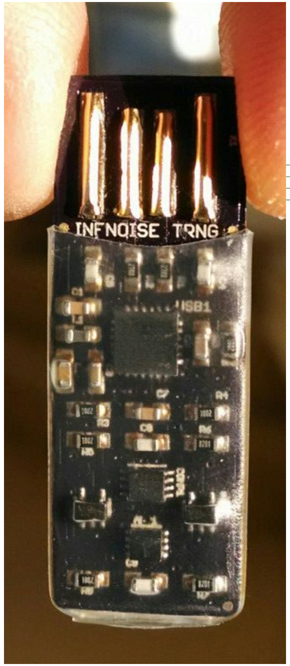
ADC sample an Avalanche Diode Noise Source (xor) with RF Energy

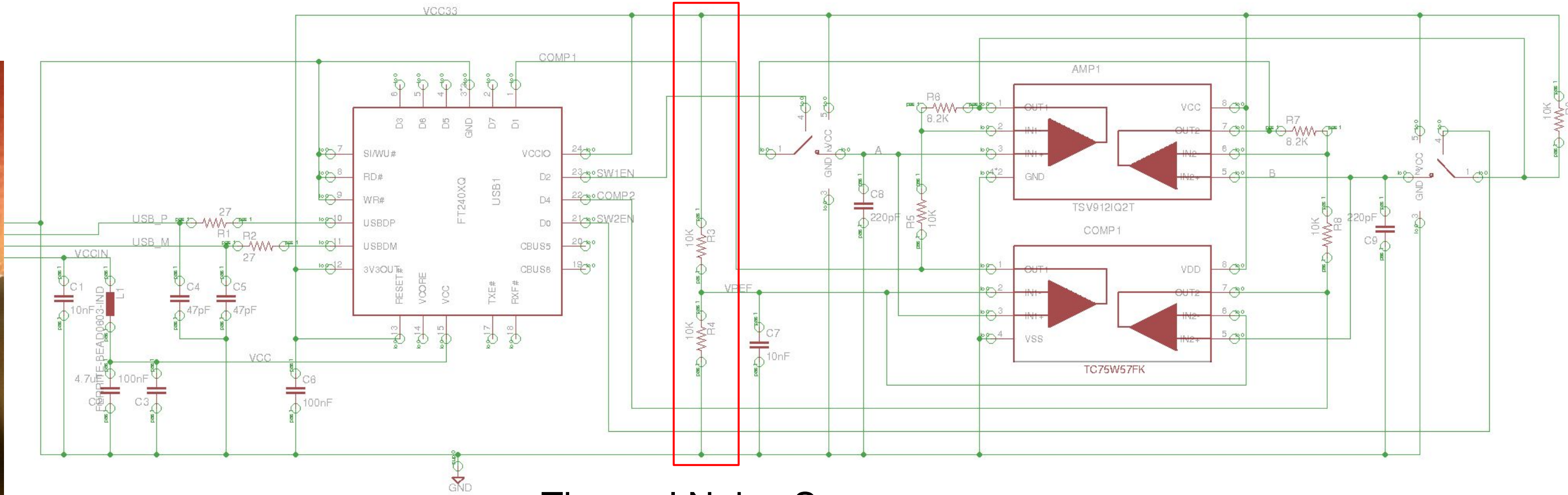
“Good” Entropy (**~.935 bits entropy/bit**)



RF Noise Source
Reverse Biased Diode Noise
+3V supply
IC1
IN
OUT
EN
BYP
GND
3.3v VREG
C1
1uf
R1
1k5
R2
22
C6
C9
47pf 47pf
32MHz
X1
C4
C12
18pf
18pf
C2
C5
C7
C10
.1uf
IC2
RESET_N
XOSC_Q1
XOSC_Q2
DVDD
AVDD
DCOUPL
RBIAS
RF_P
RF_N
DVDD_USB
USB_P
USB_N
DGND_USB
GND
EXP
CC2531
P2_4/XOSC32K_Q1
P2_3/XOSC32K_Q2
R4
56k
C11
1uf
C13
18pf
R5
1k
LED
PROG1
TAB
Programmer
DC
DD
L2
Ferrite Bead
C19
.1uf
+3V3
10uH
L1
20V generator
D1
IC3
AP3105
SHDN
VIN
SW
R6
470k
R7
27k
C15
1uf
4k7
R9
T2
T1
Avalanche diode
C17
.1uf
T3
R11
470k
R3
4k7
Amplifier
Comparator
(c) Paul Campbell 2014
TITLE: rng.1.0
Released under GPL V3.0
Document Number:
OSHW LOGO
LOGO
REV:
Date: 14/12/14 3:04 PM
Sheet: 1/1

More RNG Generators (Infinite Noise)





Thermal Noise Source

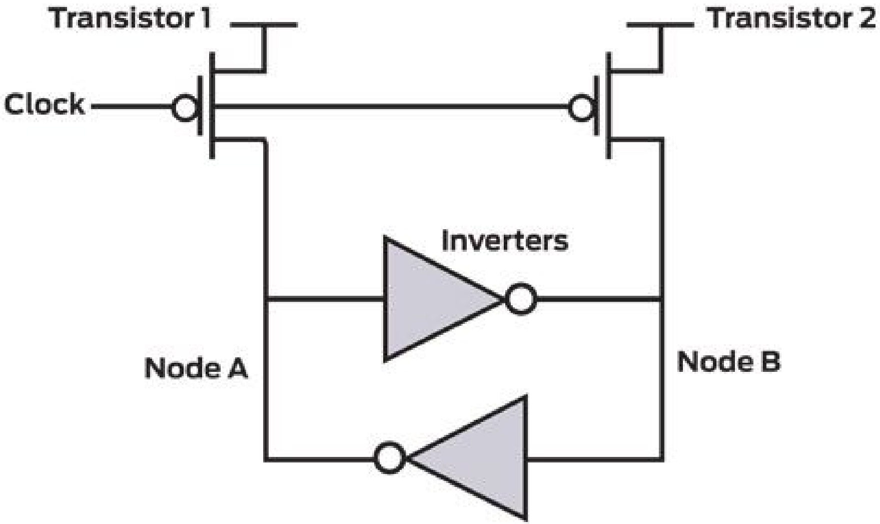
Infinite Noise:

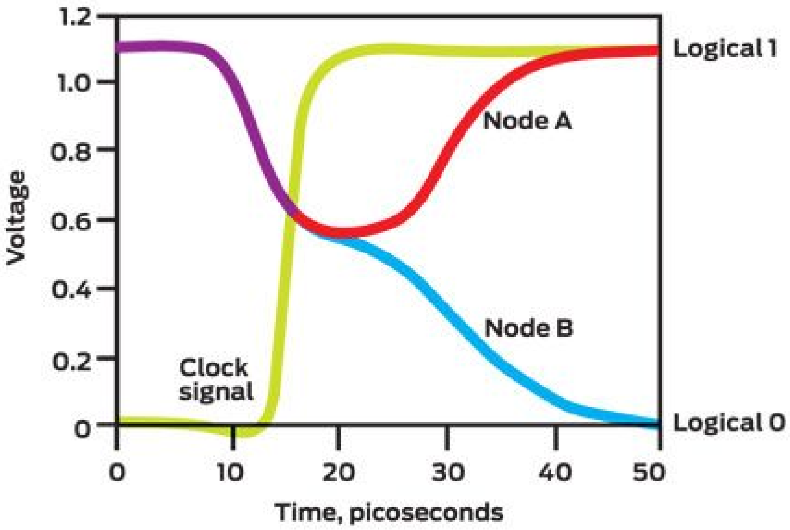
- Open source
- “All three boards should produce log2(1.82) = **0.864 bits of entropy per bit** by design”

-Entropy calculated based on loop gain of the system, amplifies resistor RMS noise voltage

https://github.com/waywardgeek/infnoise

More RNG Generators (Intel)





Intel's Latest RNG Generator
-Uses an astable set of inverters (implemented with an SRAM cell and logic)
-Moves system into an unstable region until thermal noise nudges system from equilibrium

http://spectrum.ieee.org/computing/hardware/behind-intels-new-randomnumber-generator

Many good ideas here! Can we do better?

1) Small, low cost, low power for IoT
2) Auditable entropy source
3) Can we get better entropy than other designs?

Let’s build a RNG!

The Noise source

Choice of noise is critical!

a) Probabilistic Noise
b) Large Magnitude Noise
c) Auditable
d) Cheap, made from commodity parts

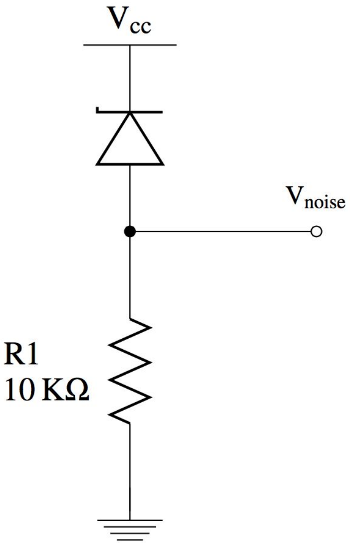
Based on these choices we chose Diode Avalanche noise as the noise source.

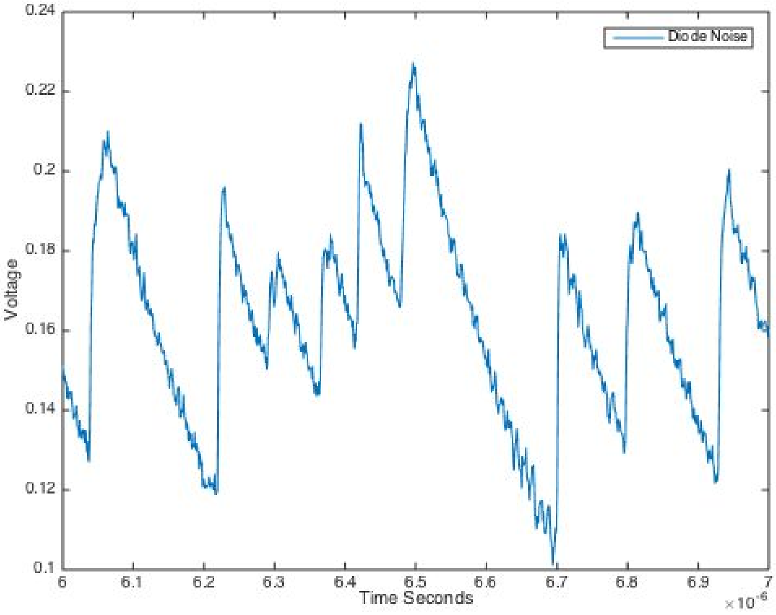
Reverse Bias Diode Noise

When reverse biased >6V, zener diodes exhibit avalanche current.

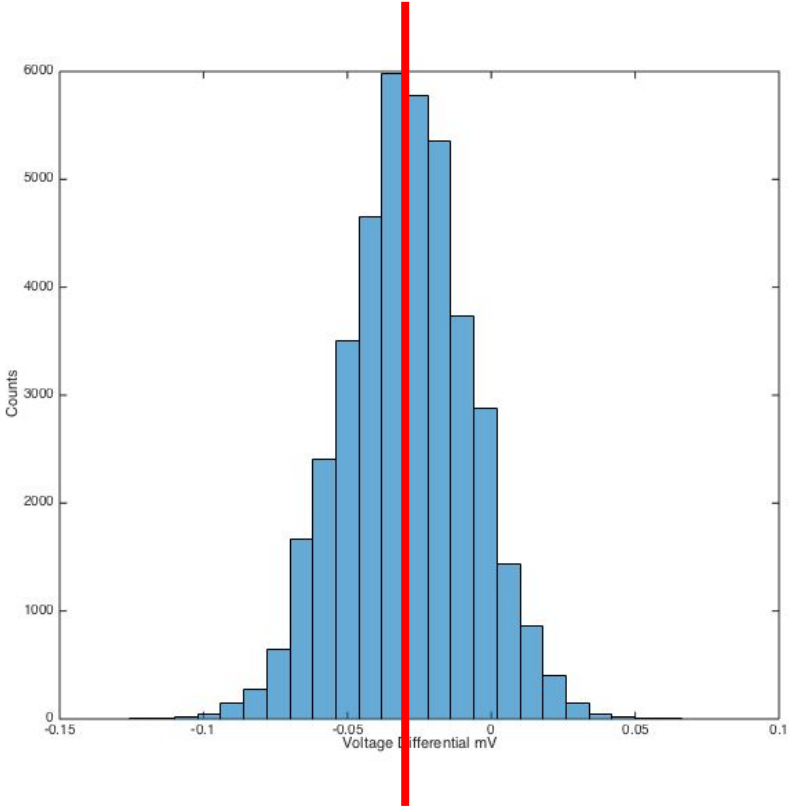
Electron multiplication as they travel across the junction.

Similar to “shot noise”, but of much high magnitude.

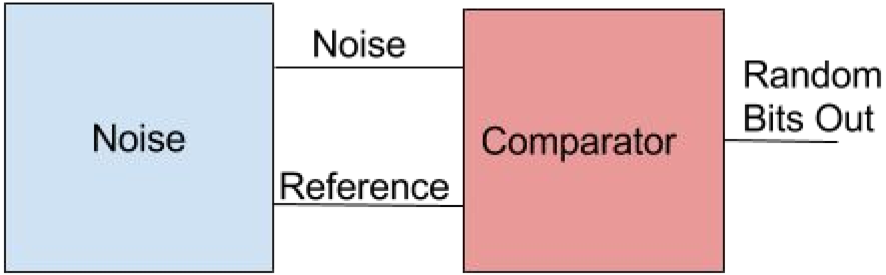




Random Bit Generator, the Naive approach:



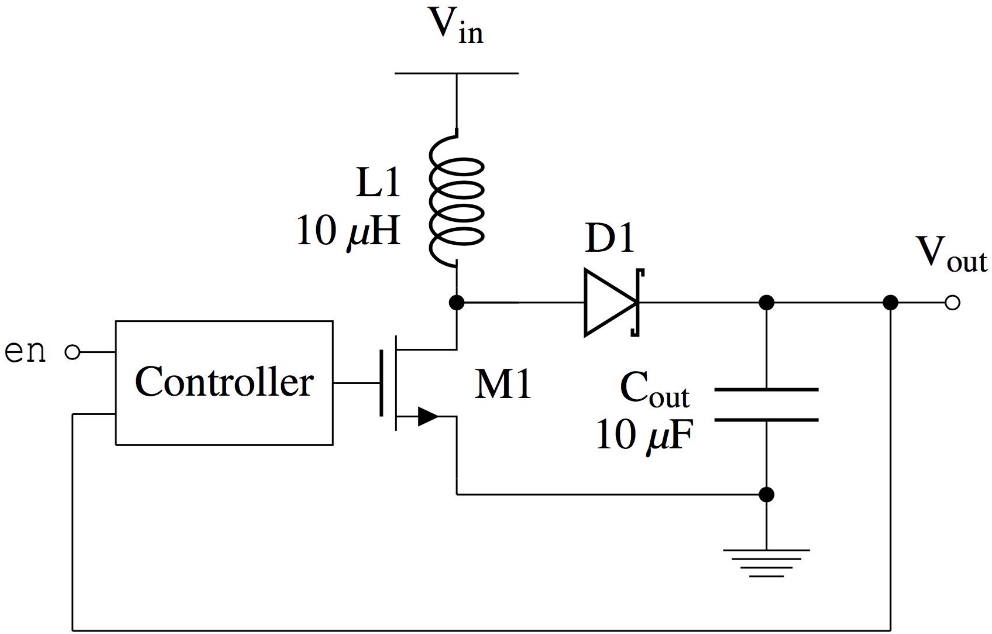
Reference Voltage



Drawbacks we need to address

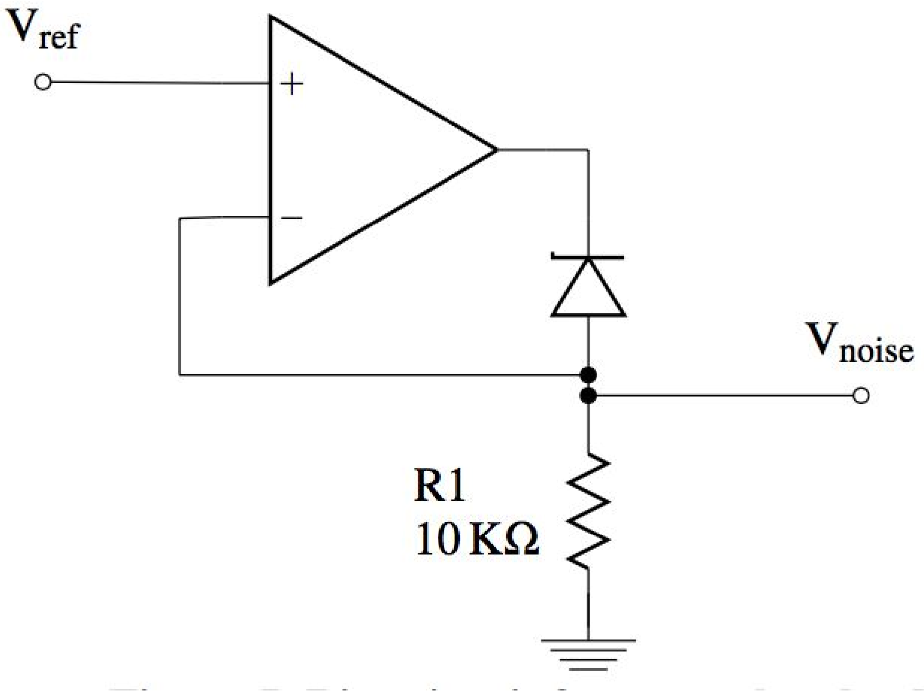
1) Requires a high voltage supply
   -Means we will need to add some type of step up converter
2) Diode drops can drift over time
   -Moves the mean of the distribution over time
   -Need some type of way to track this
3) Reference Voltage could be susceptible to noise injection
   -If reference moves, could start measuring more 1’s than 0’s, reduces entropy

1) Requires a high voltage supply - Use a boost



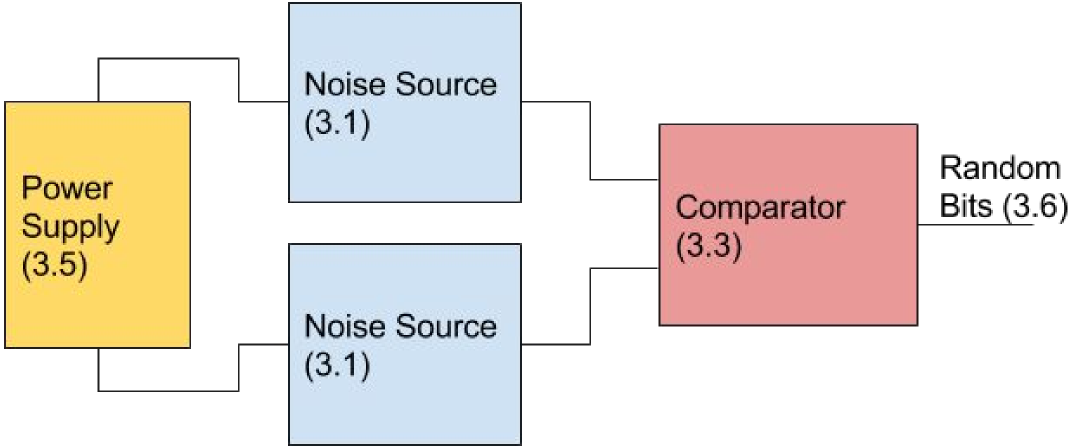
Benefits:
-Relatively cheap way to create high voltage rails (~$0.70)
-Can be toggled on and off to avoid creating switching noise

2) Handling Drift - Use Negative Feedback



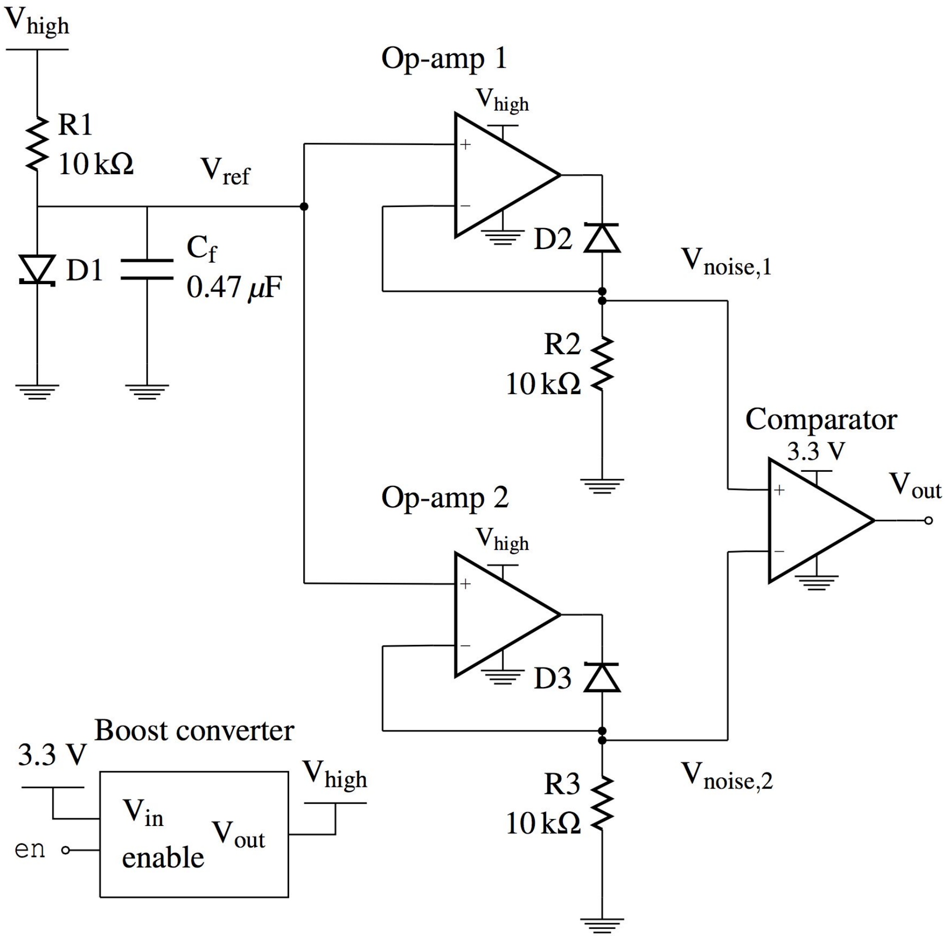
Benefits:
-DC operating point is always set w.r.t. a reference voltage
-Component variability is tolerable
-Has the ability to reject power supply noise injection

3) Reference noise immunity? Use two noise sources.

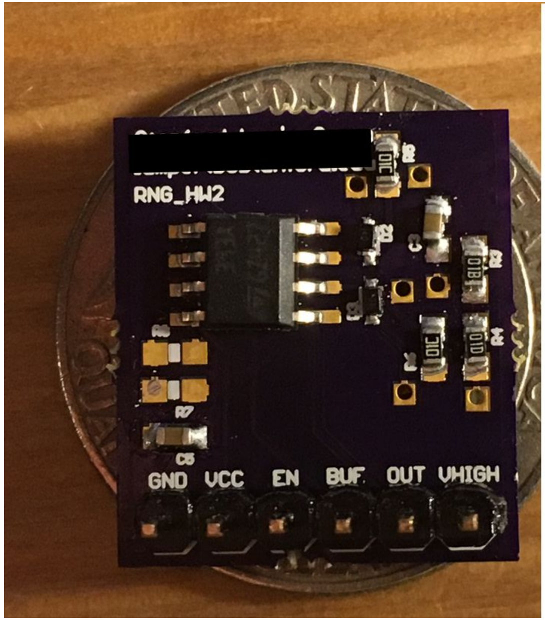


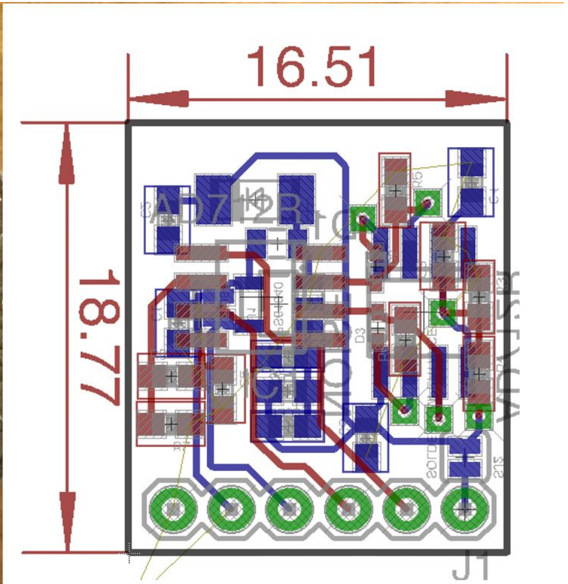
Benefits:
-Both noise sources are biased to the same mean, so comparator is only comparing the noise distributions
-Two identical noise sources experience similar noise, comparator common mode rejection helps reduce external effects.

Final Circuit



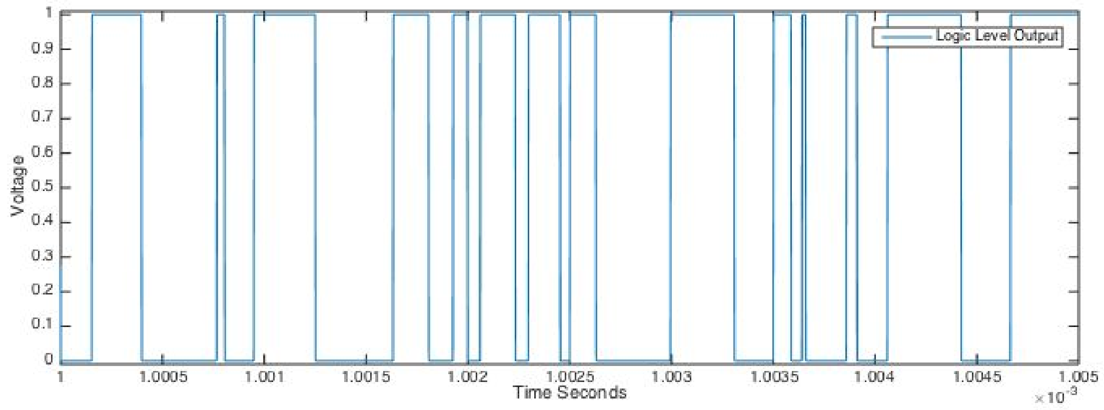
Implementation

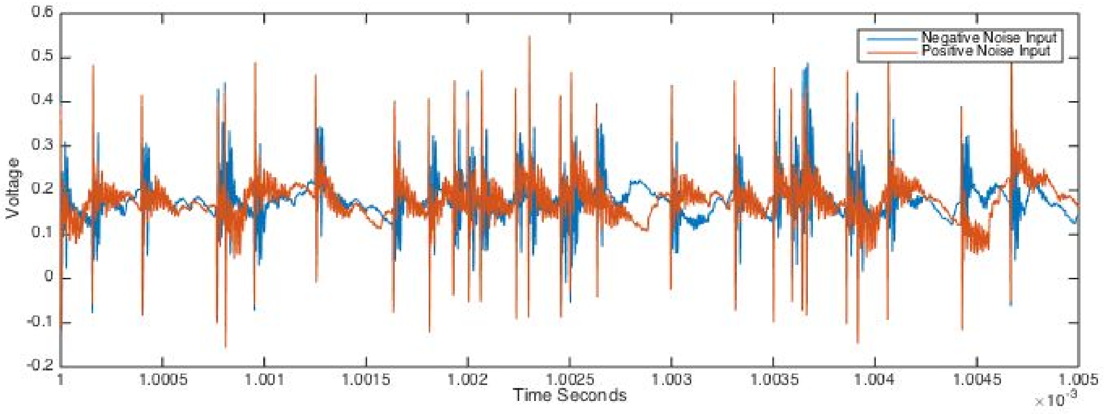




Board Area: <1.5cm^2
BOM Cost: $1.44@10k quantities

Results


Logic Level Output
Voltage
Time Seconds
×10^-3


Negative Noise Input
Positive Noise Input
Voltage
Time Seconds
×10^-3

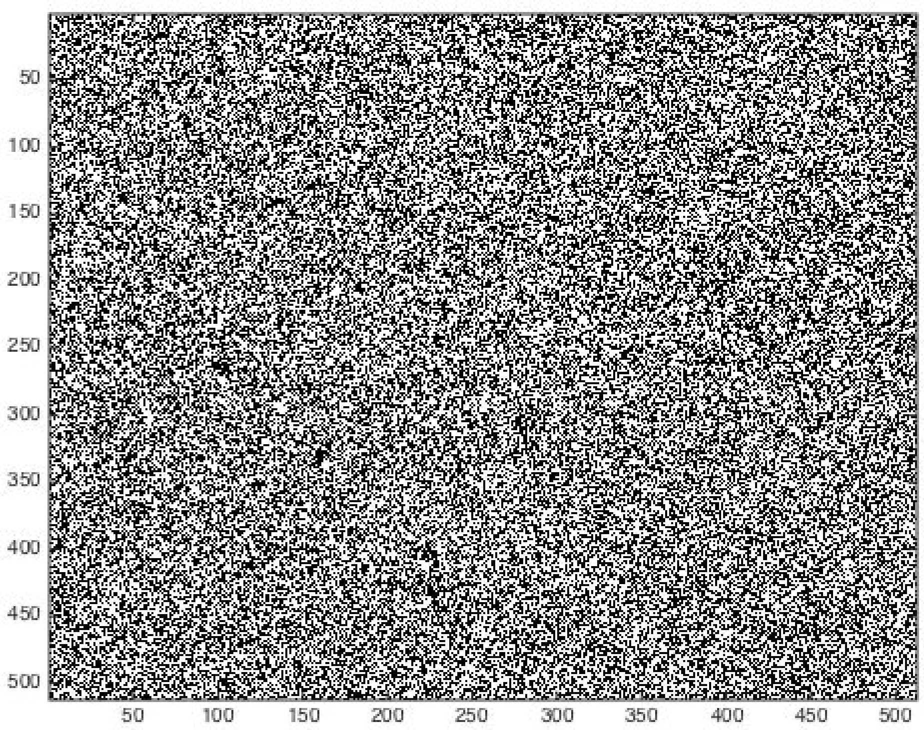
Results

Bit generation:
~6.6M Transitions/Second

Sampled bits at 128KHz to produce uncorrelated bits

<3uJ per bit (10x more power per bit than Zigbee radio)

	Entropy	Serial Correlation
–13 °C	0.991007	–0.000525
25 °C	0.995133	0.000072



Now that we have high entropy, what next?

Want to keep generating entropy bits without needing to keep powering the HWRNG

Use HWRNG to seed a PRNG (AES counter mode) [Corrigan-Gibbs,USENIX HotOS, May 2015]

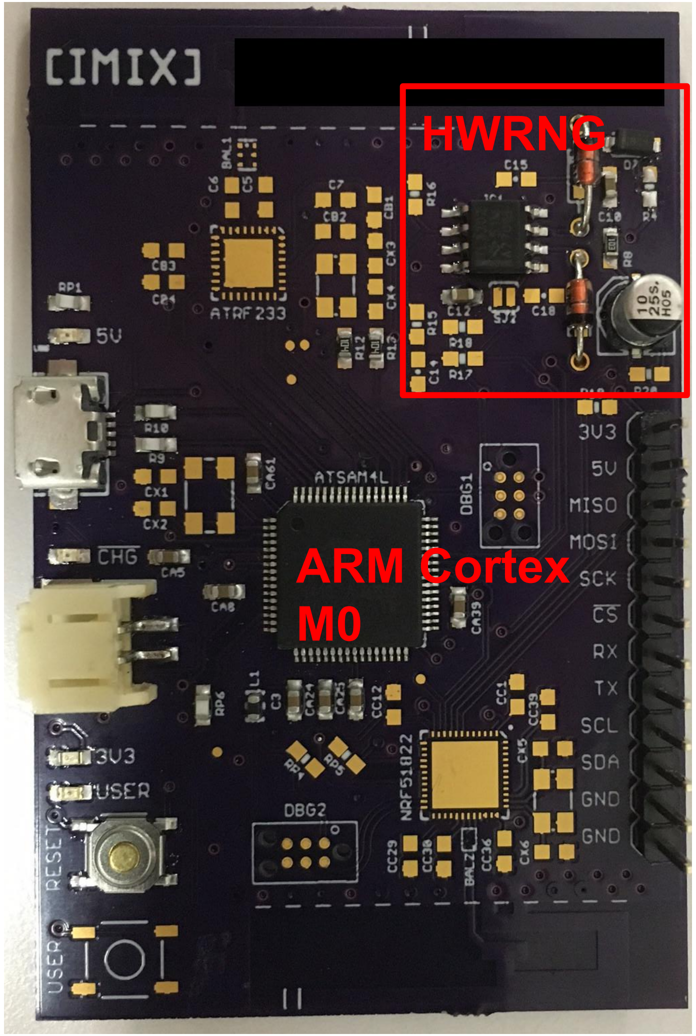
1) Sample 1024 raw bits
2) Debias using Von Neumann technique
3) Once you have sufficient entropy use a SHA256 hash to produce 256bits of entropy to seed AES in CTR mode.
4) Use AES in CTR mode and mask output to generate all future bits

Future Work

-Integration into the Imix development board

-Working on integrating this into the boot sequence to seed PRNG (AES in CTR mode)

-Raw bits still need health check, have several nodes available to do this but need to implement them



Acknowledgements

Phil Levis (pal@cs.stanford.edu)

Riad Wahby (rsw@cs.stanford.edu)

Shane Leonard (shanel@stanford.edu)

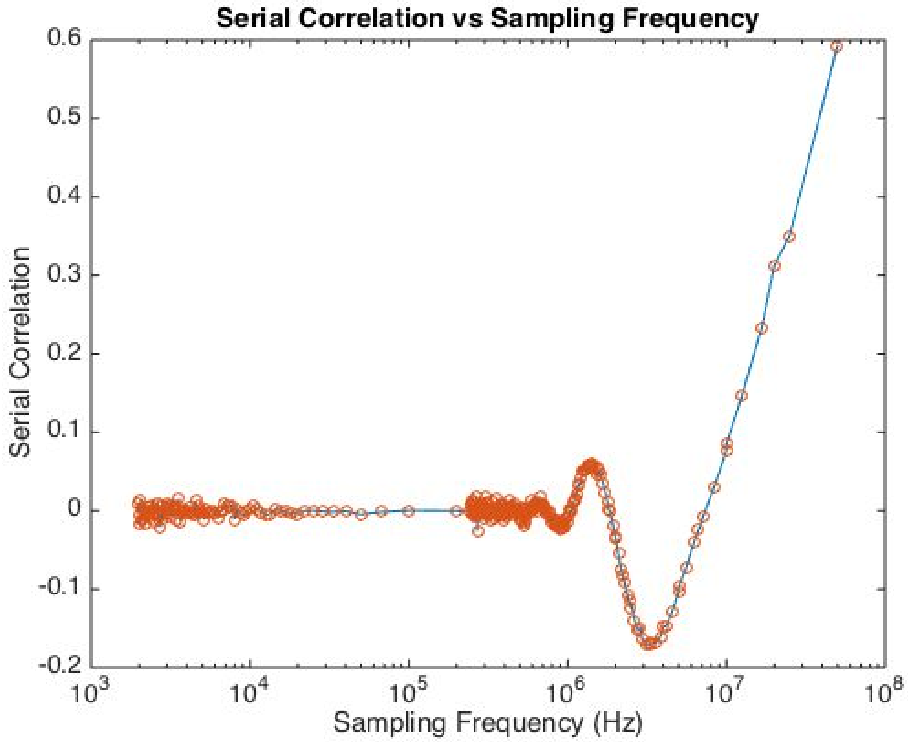
Jesse Walker (Intel)

Contact:

Ben Lampert (lampertb@alumni.stanford.edu)

Questions?

What rate to sample at?



Power Supply Toggling