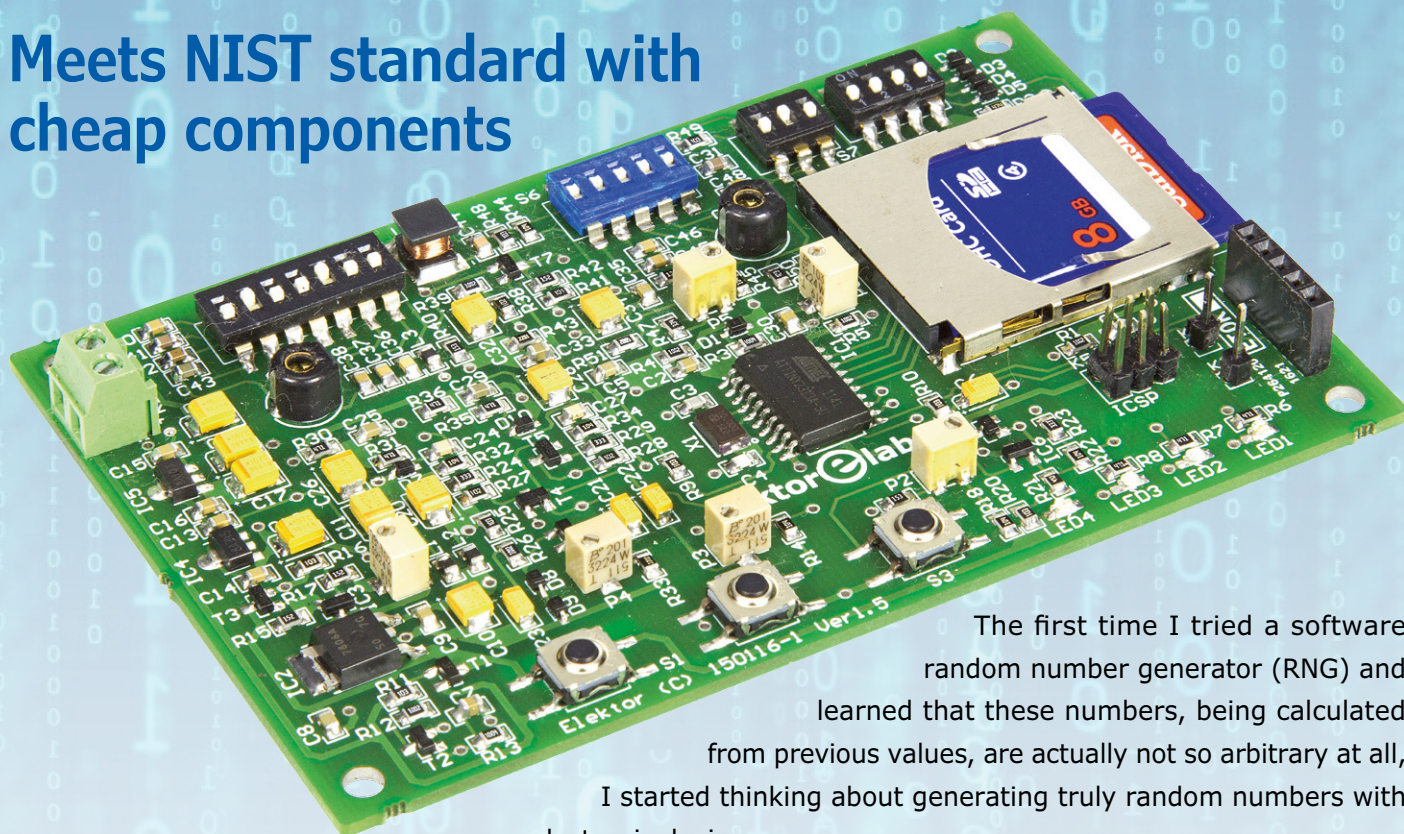# Truly Random-Number Generator

Meets NIST standard with cheap components

The first time I tried a software random number generator (RNG) and learned that these numbers, being calculated from previous values, are actually not so arbitrary at all, I started thinking about generating truly random numbers with an electronic device.

By **Luka Matić** (Croatia)

Later, when I learned the basics of cryptography, I discovered more reasons for a good Random Number Generator (RNG) to be a useful thing to have. Secure data encryption, for instance, is almost impossible without one. Gaming and gambling applications also require top-class RNGs [2], and—for the mathematically inclined—did you know that you can estimate the number π by using random numbers? Just search the Internet for 'Buffon' and 'Pi' to find out how. High-quality random signal generators are available commercially but tend to cost a lot of money, so I decided to build my own out of cheap and easily available components.

The circuit

For what follows, please refer to the schematic shown in **Figure 1**.

Because noise is random by nature, a noise source forms the basis of the RNG. The randomness of different types of electronic noise (shot, thermal, flicker, popcorn, avalanche, etc.) are well known. My RNG uses avalanche noise — which can be obtained from zener diodes — chosen because of its wide bandwidth and much higher amplitude compared to other noise sources. Two 12-V zener diodes (D7 and D10) are used to generate two noise signals. A differential amplifier built around T4, T5 and T6 amplifies the difference of both signals, to improve the chances that the noise signal is substantially random. This amplifier also removes common-mode signals due to interference and disturbances that may affect both diodes.

The next stage (T7) amplifies the noise signal further, up to 0.5 to 1 Vpp, which is enough for the analog comparator integrated in the ATtiny2313 micro. It also

serves as an adjustable corrective filter that allows the RNG to meet the quality criteria for random signals set by standardization organizations worldwide (see **inset**).

The output of the analog comparator is sampled at a rate of around 800 kHz.

Consequently, certain patterns in the random bit stream may be correlated to subharmonics of the sample rate present in the noise signal. Bit patterns 0x00 and 0xFF correspond to a subharmonic of 50 kHz; patterns 0x0F and 0xF0 correspond to 100 kHz; patterns 0x33 and

0xCC to 200 kHz; and patterns 0x55 and 0xAA to 400 kHz. Too many or too few occurrences of a certain bit pattern indicate that the frequency response of the corrective filter must be adjusted to decrease or increase the gain at the 'offending' frequency. This can be done
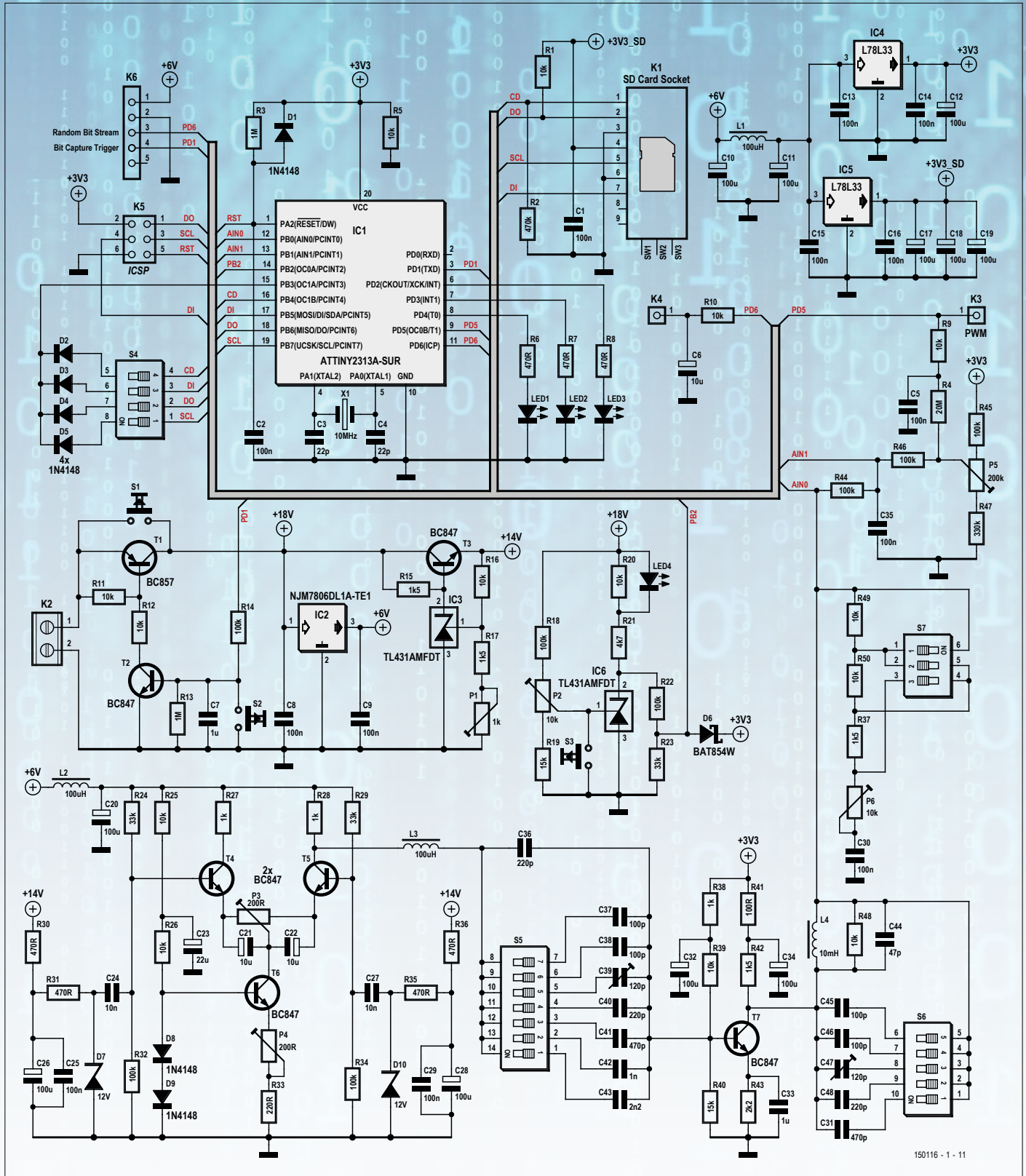

Figure 1. The Random Number Generator or RNG cleverly combines analog and digital electronics to produce a truly random bit stream.

with DIP switches S5, S6 and S7 and by adjusting trimmers C39, C47 and P6. The sampled output of the analog comparator is recorded on the SD card in one, large file. The SD card must be formatted as FAT32 (see inset). Before the RNG is switched on, DIP switch S4 must be set to define the desired size of the file, ranging from 16 MB to 3.8 GB (see **Table 1**). After a successful recording a file named 'random.hex' is created on the SD card.

The stream of random bits is also output on PD6 so that its quality can be monitored in real time with the Bit Stream Analyzer (see below), a tool that measures certain bit-pattern-occurrence frequencies — very handy for adjusting the filter around T7. To allow synchronizing to this bit stream, PD1 produces a short

COMPONENT LIST RNG

Resistors

Default: 0805, 100mW, 5%
R1,R5,R9,R10,R11,R12,R16,R20,R25,R26,R39,
   R48,R49,R50 = 10 kΩ
R2 = 470kΩ
R3,R13 = 1MΩ
R4 = 20MΩ
R6,R7,R8,R30,R31,R35,R36 = 470Ω
R14,R18,R22,R32,R34,R44,R45,R46 = 100kΩ
R15,R17,R37,R42 = 1.5kΩ
R19,R40 = 15kΩ
R21 = 4.7kΩ
R23,R24,R29 = 33kΩ
R27,R28,R38 = 1kΩ
R33 = 220Ω
R41 = 100Ω
R43 = 2.2kΩ
R47 = 330kΩ
P1 = trimpot, 1 kΩ, 11-turn, SMD
P2,P6 = trimpot, 10kΩ, 11-turn, SMD
P3,P4 = trimpot, 200Ω, 11-turn, SMD
P5 = trimpot, 200kΩ, 11-turn, SMD

Capacitors

Default: 0805
C1,C2,C5,C8,C9,C13,C14,C15,C16,C25,C29,
   C30,C35 = 100nF
C3,C4 = 22pF
C6,C21,C22 = 10µF, 6.3V, case A
C7,C33 = 1 µF
C10,C11,C12,C17,C18,C19,C20,C26,C28,C32
   ,C34 = 100µF, 10V, case B
C23 = 22µF, 10V, case A
C24,C27 = 10nF
C31,C41 = 470pF
C36,C40,C48 = 220pF
C37,C38,C45,C46 = 100pF
C39,C47 = 120pF adjustable capacitor (eBay is
   your friend)
C42 = 1nF
C43 = 2.2nF
C44 = 47pF

Inductors

L1,L2,L3 = 100µH, 0805
L4 = 10mH, 2220

Semiconductors

IC1 = ATtiny2313V-10SUR, programmed
IC2 = L7806CD2T-TR (or NJM7806DL1A-TE1)
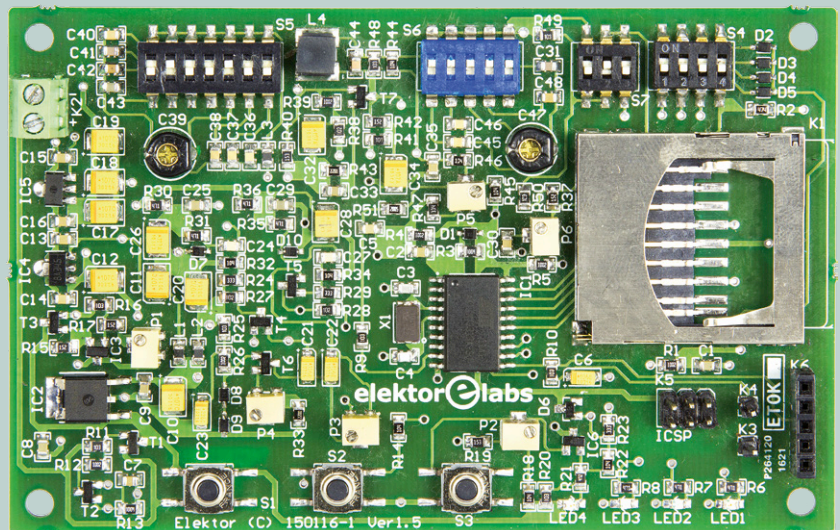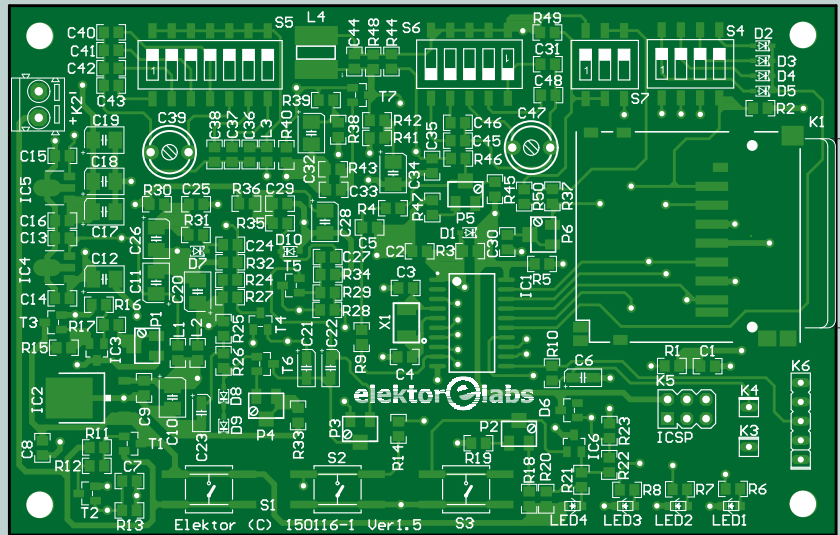
IC3,IC6 = TL431AMFDT
IC4,IC5 = L78L33ACUTR
D1,D2,D3,D4,D5,D8,D9 = 1N4148WS
D6 = BAT854W
D7,D10 = BZX384-C12
T1 = BC857CW
T2,T3,T4,T5,T6,T7 = BC847BW
LED1,LED4 = green, 2x1.25 mm
LED2 = red, 2 x 1.25mm
LED3 = yellow, 2 x 1.25mm

Miscellaneous

X1 = 10 MHz quartz crystal, 18pF, 5.0x3.2mm

S1,S2,S3 = Tactile switch
S4 = DIP switch, 4-way
S5 = DIP switch, 7-way
S6 = DIP switch, 5-way
S7 = DIP switch, 3-way
K1 = SD card connector type
   SDBMF-00915B0T2
K2 = 2-way PCB screw terminal block, 3.5 mm
   pitch
K3,K4 = 1 pinheader pin
K5 = 6-pin pinheader (2x3), 0.1" pitch
K6 = 5-way pinheader, 0.1" pitch

(<1 μs), active-low pulse at the start of every bit. These pulses are too short to accidentally power off the RNG.

Power

To avoid introducing interference sources that may degrade the quality of the RNG output, the device is supposed to be powered from two 9-V batteries in series. The 18 volts we get from them are needed for the 12-V zener diodes D7 and D10. This type of diode is known to produce a good amount of noise at a reasonable voltage.

Transistors T1 and T2 along with output PD1 act as a power-on (and auto power-off) flip-flop. Pressing button S1 switches the circuit on, making the MCU come alive so that it can activate port PD1 and keep its own pants up.

The circuit around IC6 functions as a low-battery indicator, and informs the MCU when the voltage drops to about 15.4 V. When this happens, the MCU safely terminates writing the SD card before shutting down the circuit (by means of PD1) to prevent both data corruption and excessive draining of the batteries. A low-battery condition can be simulated with S3, which functions as a soft-stop button used to stop the recording at any time while keeping the data. S2 is a hard-stop button which turns off the power immediately. Pressing it while a recording is in progress is likely to result in a corrupted file on the SD card. Use it after adjusting trimmer P5 or when something goes wrong.

The LEDs

After pressing S1 to power on the RNG, and if the battery voltage is high enough, LED4 will turn on. Then LED3 comes on, followed by LED1 to show that the SD card was correctly initialized. LED1 then turns off, meaning that the recording of the random bits has started; it will turn on again when the file of the desired size has been written and from then on it will stay on as long as the battery voltage remains high enough. LED3 will flash repeatedly during recording: a short flash at the end of each cluster written, and a long flash when the FAT table is updated. LED2 indicates if any problems occurred during SD card communication.

Bit Stream Analyzer

Generating quality random data is not easy, which is why the RNG features several adjustable resistors and capacitors

that need to be set properly. Even though it is possible to record a bit stream, analyze it, readjust the RNG, and repeat this process as often as necessary, doing so is not very practical. For this reason a special tool was developed: the Bit Stream Analyzer (BSA). With the BSA things

become more convenient as it allows you to observe the tuning effects in real time. The BSA consists of an ATMega8 and a 2x16-character alphanumeric display (Figure 2) connected to K6 of the RNG. For its input it takes the random bit stream produced by the RNG on PD6
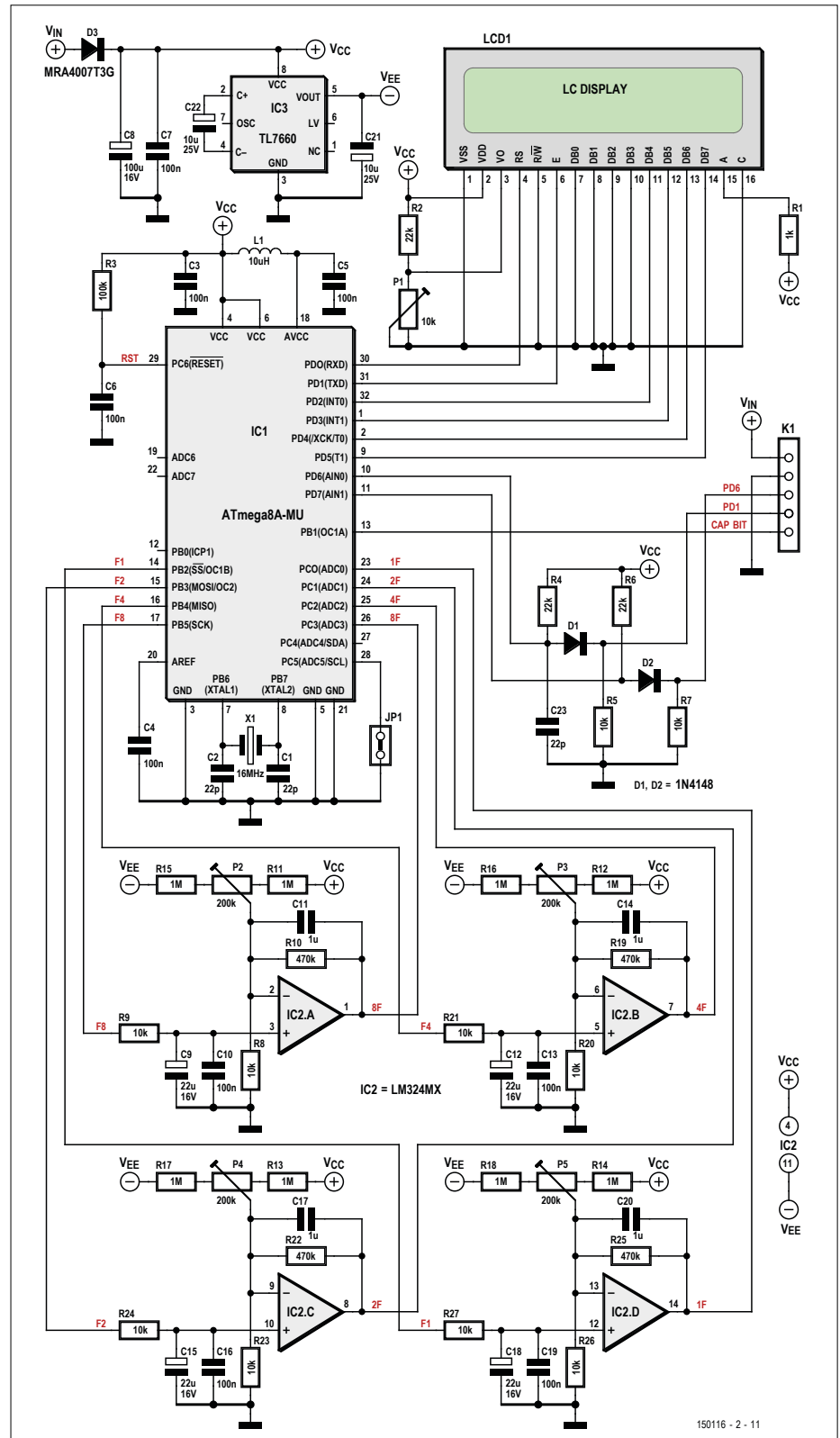

Figure 2. The Bit Stream Analyzer or BSA allows real time tuning of the RNG.

and a bit synchronization signal on PD1. The BSA is a kind of spectrum analyzer tuned to the first four subharmonics of the RNG sample rate (800 kHz). It displays the amplitudes labelled as f1 (50 kHz), f2 (100 kHz), f4 (200 kHz) and f8 (400 kHz), and it displays the 0/1 ratio.

The MCU is helped by a kind of 'analog computer' (IC2) to determine the levels of the four subharmonics because the ATmega8 is not fast enough to do all the work itself. The MCU only handles the 0/1 ratio, and 'subcontracts' the subharmonics to the analog computer. The MCU outputs a short, constant-length pulse on PB2 to PB5 for each occurrence of a characteristic pattern 0x00, 0x0F, 0x33 or 0x55 and reads the processed analog value at PC0 to PC3.

The number in the upper right corner of the display is the 0/1 ratio which should be around 512. It is controlled by P5 of the RNG (not of the BSA) and the PWM auto-offset. The four subharmonics should have magnitudes in the same ballpark (i.e. within ±10%), the exact values do not matter much.

Before the BSA can be used, the DC offsets of the operational amplifiers must be balanced:

1. Fit jumper JP1 to connect PC5 to GND and then power up or reset the BSA. The MCU will enter the Zero-offset Adjustment Mode, indicated by the character 'Z' appearing on the LCD.
2. Adjust trimpots P2 to P5 to set the values of f1, f2, f4 and f8 to a value slightly above zero; between 10 and 20 is fine.
3. Remove jumper JP1, and reset (or power cycle) the BSA.

Tuning the RNG

Building a precision device with low-cost components usually results in one or more parameters that require adjusting manually. The RNG has several of them:

Trimpot P1

Adjust to obtain 14 V at the emitter of T3.

Trimpot P2

Determines the voltage that triggers the 'low battery' warning (LED4 turns off). If the RNG is powered by two 9-V NiMh batteries, the warning should trigger at 15.4 V (because such batteries contain seven 1.2-V NiMh cells, and when discharged to 1.1 V per cell this totals to 15.4 V). Connect an adjustable power supply on K2; vary its voltage between 15 V and 16 V. Adjust P2 to make LED4 turn off when the voltage at connector K2 drops to 15.4 V.

COMPONENT LIST BSA

Resistors
Default: 0805, 100mW, 5%
R1 = 1kΩ
R2,R4,R6 = 22kΩ
R3 = 100kΩ
R5,R7,R8,R9,R20,R21,R23,R24,R26,R27 = 10kΩ
R10,R19,R22,R25 = 470kΩ
R11,R12,R13,R14,R15,R16,R17,R18 = 1MΩ
P1 = 10kΩ, trimpot, SMD
P2,P3,P4,P5 = trimpot, 200kΩ

Capacitors
Default: 0805
C1, C2,C23 = 22pF
C3,C4,C5,C6,C7,C10,C13,C16,C19 = 100nF
C8 = 100µF, 16V, radial can SMD
C9,C12,C15,C18 = 22µF, 16V, radial can SMD
C11,C14,C17,C20 = 1µF
C21,C22 = 10µF, 25V, radial can SMD

Inductors
L1 = 10µH, 0805

Semiconductors
D1,D2 = 1N4148WS
D3 = MRA4007T3G
IC1 = ATmega8-16AU
IC2 = LM324MX
IC3 = TL7660CD

Miscellaneous
X1 = 16 MHz quartz crystal, 18pF, 5x3.2mm
JP1 = 2-pin pinheader, 0.1" pitch
Jumper, 0.1" pitch
K1 = 5-way pinheader socket, 0.1" pitch (solder side)
LCD1 = alphanumeric, 16 columns, 2 lines
16-pin pinheader (for LCD), 0.1" pitch
16-way pinheader socket (for LCD), 0.1" pitch
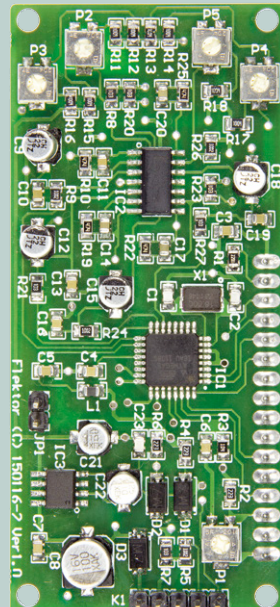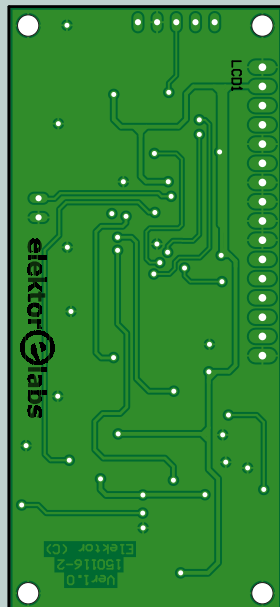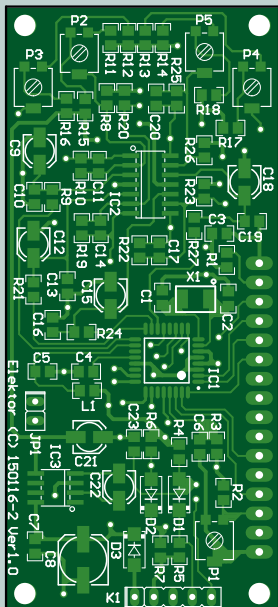
Trimpots P3 and P4

The collector currents of transistors T4 and T5 must be 1 mA. Adjust trimpots P3 (symmetry) and P4 (total current) to obtain a voltage drop of 1 V across both resistors R27 and R28.

Trimpot P5

1. Turn off the power (press S2, hard stop);
2. Hold S3 (soft stop), then press S1 (start) to enter Test Mode. LED3 turns on;
3. Connect an oscilloscope (or a DC voltmeter) to PD5 to monitor the PWM signal that compensates DC drift. Adjust P5 to make the PWM signal stabilize at around 50%;
4. The Test Mode runs for about 40 seconds, then the PWM value is written to the internal EEPROM of the MCU and LED1 is turned on. Restart the test mode if you need more time. If the PWM saturates at 0 or 100% LED2 turns on and the PWM value is not written into the EEPROM. If this happens, restart Test Mode.

S5, S6 and S7

Figure 3 shows the transfer function of the circuitry around T7. S5 together with its capacitors control the lower cut-off frequency. Use S6 to adjust the filter's resonant frequency as in:

$$f_{res} = 1 / (2\pi \times \sqrt{(L4 \times C_{S6})}) \ [Hz]$$

where S7 sets the maximum dip of:

$$20 \times \log_{10}(R_{S7} / (R_{S7} + R48)) \ [dB].$$

Auto-trimming

The last PWM value is always kept in internal EEPROM. In normal random data recording mode, the PWM duty cycle is adjusted constantly. At the end of a recording (at completion or a soft stop due to a low-battery condition) the current PWM duty-cycle value is written to EEPROM to be used the next time the RNG is powered on. It is therefore recommended to first record one small 16 MB file (and discard it) prior to recording a file of the desired size so that the PWM signal can be adjusted by the MCU. When done, set the desired size with DIP switch S4 and restart the RNG immediately. The random data will now have the highest quality because the DC offset will be properly adjusted from the very beginning.

Sanity check

After tuning, record a small 16 MB file and analyze it with a program like WINHEX to create a histogram of occurrences of all 256 possible 8-bit patterns. **Figure 4** shows a spiky, non-uniform histogram for data that's not random enough for use in serious cryptography even though its 0/1 bit ratio is correct. Tuning more carefully brings histograms similar to the one in **Figure 5** within reach.
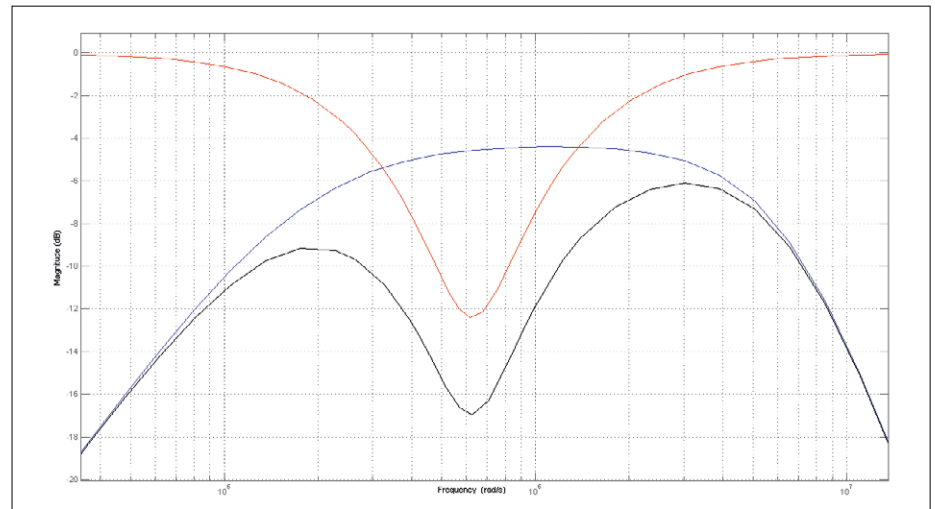

Figure 3. The simulated transfer function of the filter built around T7 (black curve). The blue curve is the transfer function of T7, defined by its bandwidth. The lower cutoff frequency is controlled by S5 and its capacitors C36 to C43. The red curve is the transfer function of the corrective filter L4-R48-CS6-RS7. For clarity, transistor amplification was left out, the real black and blue curves lie about 30 dB higher.
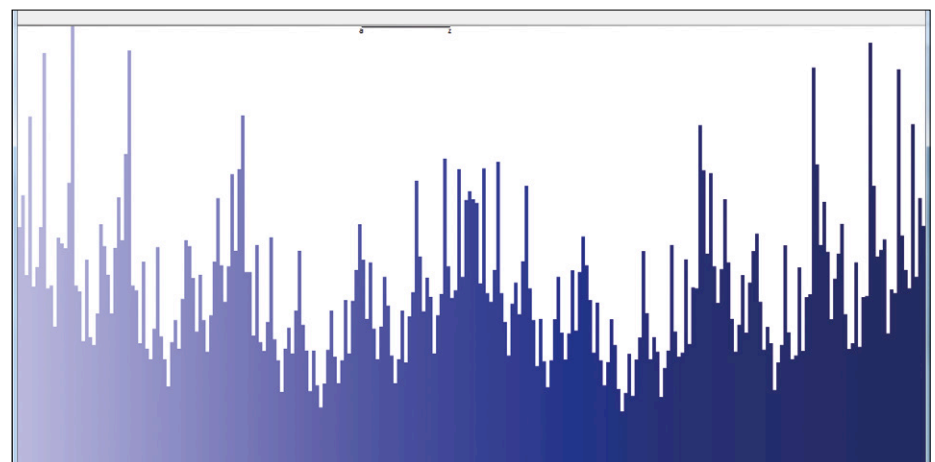

Figure 4. This histogram of the data generated by the RNG reveals a poorly tuned corrective filter. Looking only at the leftmost and rightmost bars (occurrences of respectively 0x00 and 0xFF) the zero-to-one ratio seems fine, however, because the corrective filter is not tuned properly, the occurrences of 0x0F and 0xF0 are almost three times higher than those of 0x55 and 0xAA.
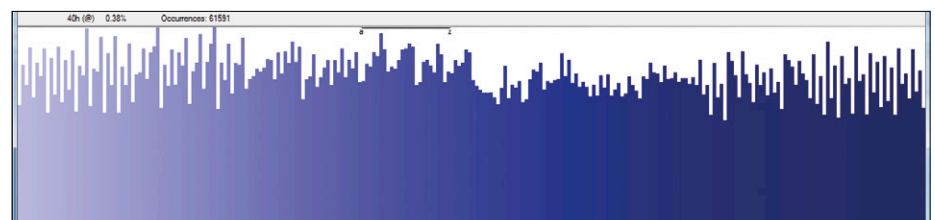

Figure 5. After proper tuning of the corrective filter the histogram is almost flat. Now you have premium-quality random data.

Beware of EMI

Please keep in mind that without proper enclosures, high-precision instruments like the RNG are sensitive to EMI and RF radiation. It is therefore highly recommended to mount the RNG inside a metal (aluminum) enclosure with the circuit Ground properly connected to the enclosure.

NIST testing

Standards to evaluate the quality of a random bit stream have been set by standardization institutes like the National Institute Standards and Technology (NIST). The NIST has defined 15 tests for randomness that boil down to three simple rules:

1. a truly random bit stream has an equal number of ones and zeros;
2. a truly random bit stream has equal frequencies of occurrences for different bit patterns;
3. no part of the bit stream is mathematically correlated to any other part of the stream.

When carefully tuned to produce histograms like the one in Figure 5, the RNG described in this article passes all 15 NIST tests with ease. The first criterion of randomness is met by care-fully adjusting the DC bias with trimpot P5. The second criterion can be met by adjusting the frequency response of the corrective filter with its capacitor and resistor banks (DIP switches S5, S6 and S7) and adjusting trimmers C39, C47, and trimpot P6). Since the source of the random bit stream is random noise, the third criterion is met automatically.

Before diving into the NIST tests, you can try a simple test with a file compression utility like 7-Zip or similar. File compression algorithms try to reduce the redundancy or correlation of data to make it smaller. Since truly random data is completely uncorrelated, it cannot be compressed. The size of the compressed file will therefore be greater than the original file, where the increase in file size is due to file format related issues.

A Windows application which performs NIST testing, pseudo-random number generators (PRNGs) that produce good pseudo-random bit streams for comparison, and sample bit streams can be found at [2].

The NIST test application requires a text file as input, hence binary files must be converted first. The application is very fussy about the exact format of the input file — only the ASCII characters '0' (ASCII code 0x30), '1' (ASCII code 0x31), and space (ASCII code 0x20) are allowed. Every binary digit must be separated by a space; leading or trailing spaces are not allowed. One mistake and the program will reject the file.

If the bit stream fails a certain NIST test, it is always related to one of the three basic criteria listed above, and some additional RNG tuning should fix it. For example, if a bit stream passes the 'Monobit' test (a simple 0/1 ratio test), but fails the 'Runs' test (i.e. counting the number of 'runs' or sequences of all ones or all zeroes) with an excessive value, then the average analog frequency is a tad too high. This can be corrected by slightly decreasing the lower cut-off frequency (by
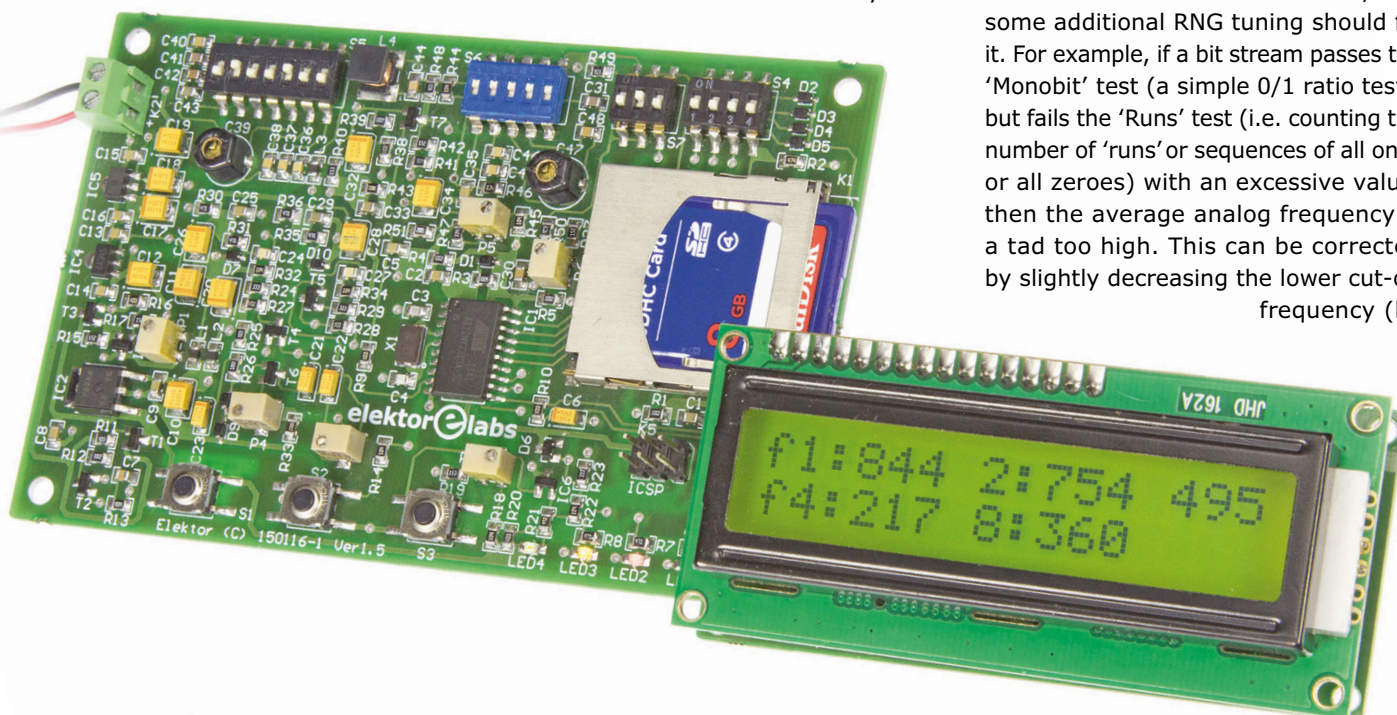
increasing the capacitor value selected with S5).

With a little tweaking of the trimmers it is not difficult to make the RNG pass all 15 (say, fifteen) NIST tests with much longer bit streams than the minimal required lengths.

Conclusion

Built with cheap, readily available components, the Random Number Generator (RNG) presented in this article is good enough to meet the standards for truly random data set by official international standardization organizations like the NIST. It shows that the DIY approach still pays off — in the true spirit of Elektor Magazine and Labs! This RNG is more than an experimental toy, it is a precision instrument. On the other hand, keep in mind that encryption schemes using it (or that use even better RNGs) can, in theory, be broken with a quantum computer, so neither Elektor nor the author can, and will, assume any responsibility for the privacy of your data — you have been warned.

(150116)

The Author

Luka Matić graduated from the FER Zagreb (Croatia) where he obtained a master's degree in automation engineering. His work experience includes automation of industrial plants, onshore as well as offshore, and of remotely operated submersible vehicles (ROSVs).

As a freelance engineer he has gained a lot of experience with all sorts of electronic devices and apparatus. Luka is not only an author for Elektor Magazine and Labs, he also co-authored a scientific article on advanced friction compensation for electrically driven positioning systems. He can be contacted at **luka.matic@gmail.com**

Web Links

[1] http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

[2] www.elektormagazine.com/labs/random-number-generator-150116

[3] www.elektormagazine.com/150116

FROM THE STORE

➜ 150116-1
PCB

➜ 150116-2
PCB

➜ 150116-41
Programmed microcontroller

➜ 150116-42
Programmed microcontroller