Noise Color Keying (NCK)

Christian Tschudin ⟨christian.tschudin@unibas.ch⟩, HB9HUH/K6CFT, Jan 2026

Textbooks on modulation typically start with introducing Frequency-Shift-Keying (FSK), then Phase-Shift-Keying (PSK), followed by all their variants. What is lost in this way of telling the modulation story is that *any* way to impress the airwaves with a suitable pattern can be used to convey information bits. "Suitable" means that demodulation should be possible with modest effort.

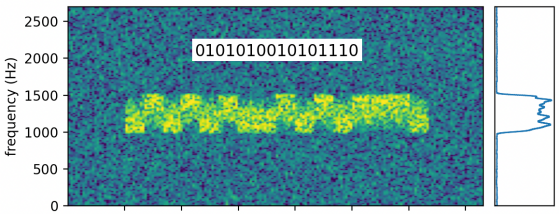

Figure 1: *Example of a noise-color-keyed signal shown in a waterfall representation.*

In this article we use *colored noise* as the basis for an unconventional modulation technique that is surprisingly easy to programmatically demodulate, namely without any Fast Fourier Transform (FFT), and based on a simple system architecture, as shown below.

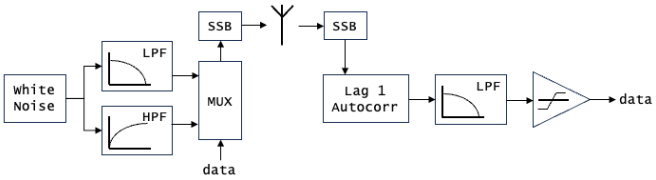

Figure 2: *Block diagram of a NCK sender and receiver.*

This report is structured as follows: After having explained our approach, we introduce "reddish" and "blueish" noise for representing a 0 or 1 value and show their complementarity, meaning that adding them creates white noise. We then provide FFT-less generation and detection methods. In a third part we report on our first evaluations based on simulations.

In a nutshell, we were able to demonstrate this novel modulation technique for transferring data packets inside a full 2.5kHz audio channel as well as a narrow "2Hz noise needle", at a keying rate of 0.2 Baud. The latter opens up the possibility to use Noise Color Keying for very weak signal communications.

1) Background and Approach

Noise by definition is a random signal: It cannot be predicted nor does it correlate with any other signal. But depending on its frequency distribution, noise can be labeled with a "color". In white noise, all frequencies are present with equal intensity. Pink noise, on the other hand, has strong low frequencies but virtually no high frequencies. The opposite is true for blue noise. The frequency distribution, that is the *weight* $w(f)$ for a specific frequency f, is defined as $w(f) = 1/f^p$ for the three colors mentioned: pink noise has $p = 1$, white noise has $p = 0$ while setting $p = -1$ results in blue noise. For the human ear, the frequency *weights* themselves are not relevant but the resulting *power* distribution is. This is also the reason why noise colors are usually explained by showing their power spectra [1]. In the upper row of Figure 3, the power distributions of pink, white and blue show up as a straight lines in a log-log plot. Note that the power spectrum is obtained from the frequency distribution by squaring each frequency's weight.

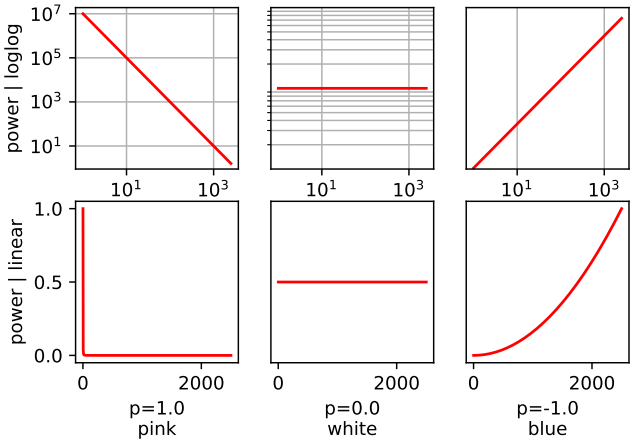

Figure 3: *Upper row: Power spectra of pink, white and blue noise, logarithmic plot. Lower row: Same spectra but shown in linear scale.*

The log-log chart is misleading though: At linear scale we see that pink noise has almost all power concentrated in the lowest frequencies while the power of blue noise is parabolically spread out (lower row of Figure 3). Ideally we want two noise colors which are *complementary*: Summing up their power would result in a flat power spectrum. In Figure 1 we showed such a modulation: Added up over the full packet's duration we get a quite flat power distribution (although we see fluctuations due to the noise' randomness). The question is: How do we have to choose the frequency distributions for obtaining two complementary noise colors?

2) Introducing "Reddish" and "Blueish" Noise

The constraint that the power spectra of two noise colors (representing the bit values F and T) should sum up to white noise means that $w_F(f)^2 + w_T(f)^2 = 1$ for all frequencies. There is a well-known pair of functions which fulfills this equation, namely the sine and cosine functions: $sin(\alpha)^2 + cos(\alpha)^2 = 1$. In other words, by shaping the frequency distribution of noise in a sinusoidal way, we get complementary noise colors, as is shown in Figure 4. We named the noise with the cosine-shaped frequency distribution "reddish" because it is heavier in low frequen-

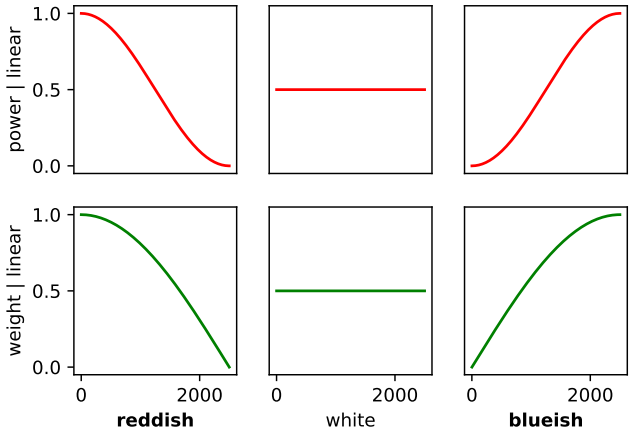

Figure 4: *Comparison of reddish, white and blueish noise using linear plots (upper row: power spectra; lower row: underlying frequency distribution).*

cies, and the sine-shaped one "blueish" as it leans toward blue noise.

3) FFT-less Noise Shaping

A straight-forward way, nowadays, to create reddish and blueish noise is to (a) generate white noise as a series of random time-domain amplitude values, (b) transform this signal to the frequency domain with FFT, (c) apply the weights as specified, and (d) transforming back to a time-domain signal via the inverse FFT.

However, there is a much simpler way to directly derive reddish and blueish noise from the time-domain samples of white noise, namely by summing and differencing two consecutive samples. What we do, in fact, is applying either a lowpass or a highpass filter.

In the case of reddish noise we filter by computing the moving average *MA(2)* of white noise wn as follows:

$$\boxed{reddish[i] = wn[i] + wn[i+1]}$$

The frequency response of a moving average filter can be found in introductory books to signal processing [2]. If M is the number of elements to be summed, the frequency response H is defined as

$$H(f) = \frac{1}{M}\,\frac{sin(\pi M f)}{sin(\pi f)},\ f \text{ in the range } (0 \ldots 0.5]$$

We use $f = \frac{F}{f_s}$ which is the discrete frequency derived from the real frequency F and the sampling frequency f_s. When inserting *M=2* into above equation and using the "double-angle property" $sin(2\alpha) = 2sin(\alpha)cos(\alpha)$, we get the desired

$$w_{reddish}(f) = \frac{1}{2}\,\frac{2sin(\pi f)cos(\pi f)}{sin(\pi f)} = cos(\pi f)$$

By symmetry, when differencing white noise as follows

$$\boxed{blueish[i] = wn[i] - wn[i+1]}$$

we get the desired frequency response

$$w_{blueish}(f) = sin(\pi f)$$

We have thus shown that the low- and highpass filtering as expressed in the algorithms (highlighted by a surrounding box) produce the desired sinusoidal frequency distributions.

4) FFT-less Noise Color Discrimination

A well-known paper on noise color identification is by Riley and Greenhall from 2004 [3]. It shows how to derive, for an interval of N values from a discrete times series, a value r_1 that is indicative for a signal's dominant noise color. Riley and Greenhall looked at phase and (single) frequency data from clocks, but the approach also works for time-domain values. Their **"lag 1 autocorrelation"** computation, which is easy to implement, is given as

$$\boxed{r_1 = \frac{\sum_{i=1}^{N-1}(x[i]-\bar{x})(x[i+1]-\bar{x})}{\sum_{i=1}^{N}(x[i]-\bar{x})^2}} \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x[i]$$

The observation is that colored noise signals have different "self-similarities", which is what r_1 captures. For example, white noise is only self-similar to itself: A copy of white noise that is shifted by any non-zero number of time steps will (ideally) have zero similarity to the unshifted version. Colored noise signals, however, have different self-similarities where it suffices to compared the signal with a copy of itself that is shifted by a single time step ("lag 1").

Pink noise is *more* self-similar than white noise (hence positive r_1 value) because the time-domain values don't differ much from sample to sample due to the dominance of low frequencies. Similarly, blue noise is *less* self-similar than white noise (negative r_1 value) because consecutive sample values differ strongly due to the dominance of high frequencies.

It turns out that Riley and Greenhall's "identification signal" r_1 also works for noise colors that do not follow the $1/f^p$ frequency distribution, specifically for our reddish and blueish noise. As reported by Riley and Greenhall, 20 to 30 samples are sufficient to identify the dominant color.

5) Evaluation

We implemented the described Noise Color Keying modulation with above FFT-less methods and verified its viability via simulation. For example, Figure 5 shows the correspondence between the input bits (red) overlayed to the r_1 signal (blue) recovered after transmission over an AWGN channel.

In a more systematic way we also studied the performance under different SNR conditions, see Figure 6. Unlike FSK or PSK, there is no sufficiently high SNR value above which transmission becomes virtually error-free: For any reasonable packet size there will be some flipped

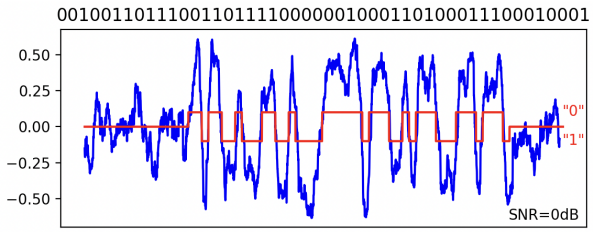

Figure 5: *Red: input to NCK modulator. Blue: retrieved autocorrelation value r_1 after traversing a noise channel.*

bits with very high probability. This means that NCK only can operate in conjunction with Forward Error Correction (FEC). In our evaluation we used 174-bit packets with FT8's LDPC encoding. The resulting curves show the usual FER behavior also found for other FEC schemas.

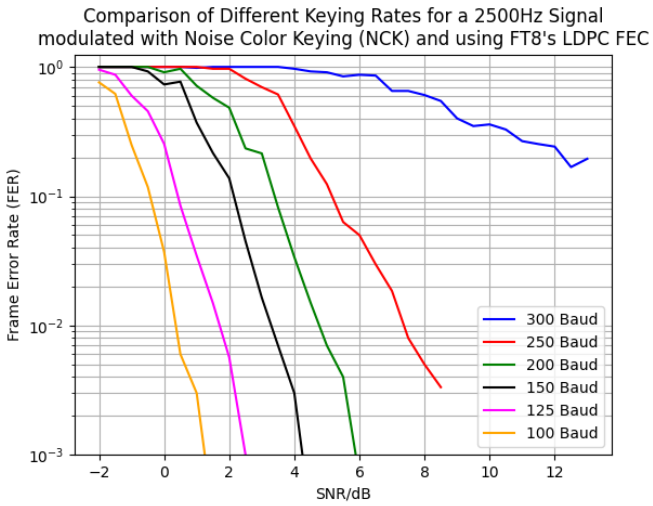

Figure 6: *Frame Error Rate vs SNR*

As expected, we observe a capacity limit that is dependent on the keying rate. A too high keying rate translates into too few sample values per symbol such that the $r1$ algorithm cannot identify the correct noise color. We further observed that lowering the keying rate allows to operate NCK in the sub-zero SNR range. The reason for this is that longer symbol times enable the r_1 algorithm to "accumulate" enough noise such that its color becomes identifiable.

6) Discussion

Noise Color Keying is less efficient (has higher E_s/N_0) than FSK where all energy is concentrated in single frequencies. This also means that FSK can be more reliably detected after traversing a noisy channel when compared to the set of randomly selected and spread out frequencies of colored noise. However, FSK is more vulnerable to birdies that will derail the recovery of single frequencies. Instead, NCK looks at the frequency distribution as a whole. Whether this NCK property is also helpful for coping with fading still needs to be investigated.

We point out that the r_1 autocorrelation signal can be used to track frequency shifts due to Doppler effects: Frequency drifts away from the NCK noise' center frequency result in shifting the r_1 curve up or down when observed over multiple symbol times. With a suitable lowpass filter, these shifts can be used to correct the receiver's center frequency.

While above FER-vs-SNR evaluation was done for a full 2.5kHz noise signal, narrow-band NCK signals are also viable, as is shown in Figure 1 for a 500Hz signal. In order to explore this path even further, we successfully demodulated a NCK signal that is just 2Hz wide, using a keying rate of 0.2 Baud and operating at 5dB SNR (in 2Hz). This is reminiscent of [4] where, in one case, a 0.125Hz-wide signal at 0.125 Baud was sent over stable VLF paths and coherently demodulated despite the estimated transmit ERP being in the μWatt range. NCK is a non-coherent demodulation method and therefore less dependent on the channel's and the receiver's clock stability: Using such very narrow signals open up the possibility to use NCK for weak signal communications in HF and to potentially compete with JT9, JT65 and WSPR while using a simpler demodulation technique.

But beforehand, the challenge of turning NCK into a packet radio format is to find robust frame detection and clock synchronization methods. FT8 spends more than 35% of its time and energy to get this part right, as otherwise all demodulation is in vain. Suggestions are welcome and are hopefully as simple as the r_1 noise identification procedure.

Source code (Python) and additional documentation for NCK can be found at `https://github.com/tschudin/NCK`.

References

[1] Wikipedia, "*Colors of Noise*." [Online]. Available: https://en.wikipedia.org/wiki/Colors_of_noise

[2] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing, 2nd edition*. California Technical Publishing, 1999, page 280. [Online]. Available: https://ia801301.us.archive.org/23/items/GuideToDigitalSignalProcessing/Guide%20To%20Digital%20Signal%20Processing.pdf

[3] W. Riley and C. Greenhal, "*Power Law Noise Identification Using the Lag 1 Autocorrelation*," 2004. [Online]. Available: https://wriley.com/Paper125Preprint.pdf

[4] P. Nicholson, "*EbNaut Coherent BPSK - Technical Details*," 2018. [Online]. Available: https://web.archive.org/web/20250718124926/https://www.abelian.org/ebnaut/