Title: Detecting Environmental Electromagnetic Patterns via Time-Compressed Temperature Sonification

Author: Troy McQuinn

Abstract:
This study explores a novel method of environmental sonification using temperature data collected from a USB thermometer. By converting these values directly to audio samples and playing them back at 8 kHz, subtle patterns emerged in the data—some of which closely resemble speech-like formants. Upon further analysis and cross-referencing with solar and geomagnetic activity logs, a consistent correlation between structured audio phenomena and known space weather events was discovered. The study suggests that this setup may be passively sensitive to low-frequency electromagnetic field variations, effectively turning the thermometer into an unintended EMF detector. This document outlines the methodology, validation against independently generated audio, and proposes future work in signal analysis, instrumentation, and citizen science.

Keywords:
Sonification, Geomagnetic Storms, Electromagnetic Interference, USB Thermometer, Time Compression, Data Sonification, Environmental Monitoring

1. Introduction

This project started with something simple: a USB thermometer quietly logging temperature every five minutes. The idea was to compress time—dramatically—by turning each reading into a 16-bit audio sample and playing them back at 8 kHz. What came out of that process wasn't just random noise or abstract patterns. Some sections sounded structured—almost speech-like.

That alone would have been intriguing, but it led to more questions than answers. Was this just a quirk of the conversion code? Could it be an artifact introduced by the way PHP handles binary packing? Or was it possible that the thermometer, sitting in a real-world environment filled with electronics and background EM fields, was picking up more than just temperature?

To rule out software artifacts, the conversion process was rebuilt in Python using the same data and logic. While the resulting audio files didn't match the original PHP output byte-for-byte—owing to differences in how each language handles binary packing and rounding—the same patterns, pitches, and structures were clearly audible. This confirmed that the observed features weren't quirks of the implementation, but genuine characteristics of the underlying data.

A more surprising discovery followed: several structured segments coincided with known solar and geomagnetic events, including the G2 storm of May 2018. It appeared that this low-cost sensor—perhaps due to poor shielding or unintended sensitivity—might be responding to environmental electromagnetic variation.

The following sections detail the data collection, audio conversion, and comparative analysis. It compares the resulting audio with control datasets and explores whether what we're hearing is just noise, or something more revealing about how digital sensors interact with their electromagnetic surroundings.

2. Methodology

2.1 Data Collection

Temperature readings were collected using a TEMPer USB thermometer (model ID: TEMPer1F), plugged into a Raspberry Pi B+ via USB. A cron job triggered a custom shell script every five minutes. This script queried the device and appended a line to a `.dat` file in the format:

```
<UNIX timestamp> <temperature (F)> <MM/DD/YY> <HH:MM:SS>
```

The dataset includes several years' worth of logs, sampled at a fixed interval of 300 seconds (5 minutes) per entry. A small number of time skips were identified (e.g., due to reboot, power outage, or USB dropout) and accounted for during processing by referencing actual UNIX timestamps.

2.2 Time Compression and Audio Mapping

Each temperature sample was directly mapped to a single 16-bit signed audio sample. The mapping was performed without interpolation or resampling. This means that one 5-minute temperature reading equates to a single point in the final audio waveform. With an output playback rate of 8,000 Hz, this produces a time-compression factor of 2.4 million. For full code examples, see Appendix E.

2.3 Sample Scaling and Clipping

To ensure signal range compatibility with 16-bit PCM audio, temperatures were clipped to a usable range of 40.0°F to 110.0°F. Values outside this range were truncated, and all remaining values were linearly scaled to fit the range [-32768, 32767]. This preserved the relative dynamics of the original signal while eliminating extreme outliers.

2.4 Audio Output Format

Raw 16-bit PCM data was written to a WAV file using custom code. The original implementation was written in PHP (using `pack()` and file streams), while the validation implementation was done in Python using `wave` and `numpy`. Both implementations produced structurally identical audio output for the same input dataset. While not byte-for-byte identical—due to slight differences in sample rounding and binary formatting—the resulting audio contained the same perceptual patterns and features.

The resulting audio was monophonic, with no fade-in, windowing, or spectral shaping applied. Any post-processing (such as filtering or pitch shifting) was done only during analysis in Audacity or via SoX command-line tools.

2.5 Timing Alignment

To cross-reference the audio with external datasets (e.g., solar activity), a complete mapping was generated between sample index and real-world timestamp. This included gap compensation where necessary. Each sample's datetime was computed either by assuming a regular 5-minute cadence or by walking through the actual timestamp list with skip handling logic.

3. Code and Hardware

3.1 Hardware Setup

The data was collected on a Raspberry Pi B+ equipped with a passive aluminum heat sink and no active cooling. This board ran a lightweight Linux environment and handled both data logging and USB communication. The temperature sensor used was a TEMPer USB thermometer, model TEMPer1F. While this device is intended for basic temperature monitoring, it appears to exhibit unexpected sensitivity to electromagnetic environmental changes, possibly due to poor shielding or circuit design.

Nearby devices within a one-meter radius include:

- A pair of powered computer speakers
- A Wacom tablet (which emits a low-level electromagnetic field for pen tracking)
- A workstation computer (typically running ~16 hours per day)
- A network switch and modem (always on)
- Two rack-mounted power conditioners

These factors are important to note when evaluating potential electromagnetic interference in the dataset.

3.2 Original PHP Implementation

The first implementation of the conversion pipeline was written in PHP. The core of the logic handled the following:

- Reading `.dat` log files line-by-line
- Parsing temperature values and converting them to scaled integers
- Packing these as little-endian 16-bit signed values using `pack()`
- Appending to a raw WAV file with a manually constructed header

This was executed as part of a weekly cron job and included no additional audio processing beyond the sample scaling.

3.3 Validation with Python

To rule out implementation artifacts, the entire process was independently recreated in Python using `numpy` and the standard `wave` module. The Python pipeline included:

- Reading parsed temperature samples into a NumPy array
- Applying the same scaling and clipping logic as the PHP version
- Writing the scaled data to a 16-bit WAV file at 8000 Hz

Although the WAV files did not match byte-for-byte due to differences in sample rounding and binary packing, the audible features and waveform shapes were consistently preserved. (see Figure 7 for waveform comparison. See Appendix E for code examples.)

4. Audio and Spectrogram Analysis

Once the audio was generated, it was analyzed both subjectively (by listening) and visually using spectrograms. Audacity was used for audio review, and SoX and Python (with matplotlib) were used for spectrogram generation.

4.1 Key Observations

- Several segments of the audio exhibited **speech-like cadence and structure**, including what appeared to be formant bands, harmonic spacing, and even quasi-syllabic modulation.
- These features were *not* consistent with random noise and survived multiple types of filtering, including bandpass (300–3400 Hz), noise reduction, and dynamic range compression.
- Most structured audio patterns were observed during the **Spring and Summer months**, while **Fall and Winter periods tended to be louder and more chaotic**, often resembling environmental or electrical noise.

4.2 Example Segments

Two segments were identified early in the analysis as being particularly structured:

- **Segment 1:** Sample range 251745 to 306937 (May–Nov 2018) (see Figure 1)
- **Segment 2:** Sample range 353963 to 408874 (Apr–Oct 2019) (see Figure 2)

Both of these aligned with periods of known geomagnetic and solar activity.

4.3 Spectrogram Characteristics

- Spectrograms of these segments revealed **repeating harmonic structures** centered between ~300 Hz and ~1500 Hz.
- In some cases, strong transitions occurred during geomagnetic storm windows—suggesting a causal or at least coincidental environmental influence.

5. Geomagnetic and Solar Correlation

Once time alignment between audio samples and real-world timestamps was established, the next step was to compare these windows against known geomagnetic and solar activity data. Publicly available datasets from NOAA's Space Weather Prediction Center and other sources were used to identify the timing and intensity of solar flares, coronal mass ejections, and geomagnetic storms. (visualized in Figure 6)

5.1 Alignment Findings

- **Segment 1** overlapped with a G2-class geomagnetic storm in May 2018 and a G3-class storm in August 2018. (see Figure 4)
- **Segment 2** contained the strongest geomagnetic storm of 2019 (G2, late August–early September) and several notable M-class flares in May of that year. (see Figure 5)
- A sharp, audible pulse in the audio was observed at sample 499736, which aligns with a G1–G3 geomagnetic storm window in early October 2021. (illustrated in Figure 3)

5.2 Seasonal Trends

Beyond individual events, there was a noticeable seasonal rhythm to the signal. Spring and summer months consistently featured more structured and coherent audio, while fall and winter recordings tended to be noisier and more chaotic. This pattern may relate to both natural geomagnetic behavior and seasonal variations in indoor electronics usage.

5.3 Interpretation

Although the phrase “correlation does not confirm causation” has become something of a formality in observational studies, it’s worth restating here: the consistency across multiple years and event windows suggests that geomagnetic disturbances are likely influencing the sensor data. Whether through direct EMF interference, subtle shifts in ambient temperature, or voltage reference instability, the thermometer appears to be acting as a crude—but revealing—environmental antenna.

It’s also worth considering the duration of these effects. Geomagnetic storms are known to produce lingering impacts on the magnetosphere, ionosphere, and even in local ground currents. These can persist for hours or days after the main event. In low-cost, poorly shielded sensors, such as the TEMPer1F, these post-storm conditions may manifest as extended fluctuations in output—possibly corresponding to the length or character of some of the speech-like “words” heard in the sonified data. While speculative, it’s plausible that what we perceive as cadence in the audio may map directly to how long these post-storm “echoes” play out in the environment.

6. Simulated Data Comparison

As a control experiment, a synthetic temperature dataset was generated covering the same time range as the original logs. The simulated data incorporated detailed seasonal trends, daily cycles, and lifestyle-based influences—such as presence in the room, sleeping hours, computer usage, and even business trip absences. Environmental noise was also introduced in an attempt to replicate the subtleties of the original.

The resulting audio, while containing recognizable and explainable structures (such as day-night cycles or weekend patterns), lacked the speech-like features observed in the real data. Even when additive noise and microfluctuations were introduced, the synthetic audio remained notably more uniform and lacked the same kind of mid-frequency harmonic clustering present in the original.

6.1 Advanced Simulation Attempts

To further test the hypothesis that pareidolia alone might explain the structured perception, a series of increasingly complex synthetic signals were produced. These included:

- **Frequency and amplitude modulation** to simulate syllabic cadence
- **Layered band-passed noise** centered on known vocal formant bands (400–2500 Hz)
- **Pitch glides and envelope variation** to mimic vocal inflection
- **Rhythmic silences** representing word or phrase breaks
- **Chaotic modulation patterns** derived from the logistic map to destabilize timing
- **Transient consonant-like bursts**, modeled as short, high-frequency filtered noise

Even with these components, the resulting audio—though sometimes evocative of vowel sounds (“ahh,” “ee”) or mechanical whispering—never achieved the perception of **complete or intelligible speech-like fragments**. No “words” or convincing phoneme chains were perceived during repeated, attentive listening.

6.2 Interpretation

This outcome suggests that while synthetic modeling can replicate the **acoustic signature** of structured temperature data, it fails to produce the **cognitive illusion of speech** unless some other, real-world nonlinear or stochastic variable is present. Such variables may include:

- Low-level EMF coupling with internal sensor timing
- Interference-induced jitter from USB polling or voltage instability
- Hardware-level nonlinearities not captured by software simulation

This further supports the interpretation that the real data contains subtle—but meaningful—interactions between the thermometer’s circuitry and its electromagnetic environment, beyond what normal temperature variance or artificial modeling can account for.

7. Discussion

A comparative experiment was performed to evaluate whether geomagnetic flux data—specifically the Kp index—could produce similar sonified audio structure when processed in the same manner as the temperature logs. The geomagnetic data was successfully converted to audio using the same 16-bit PCM mapping technique. However, the resulting audio lacked the structured, speech-like qualities present in the temperature-derived version.

This finding allows us to reasonably rule out geomagnetic indices, at least in this direct form, as the primary cause of the vocal-like characteristics in the sonified temperature data. It further supports the hypothesis that the structure is either due to localized EM interference, internal sensor behavior, or complex interactions not captured by public geomagnetic flux indices.

The results of this project blur the line between conventional data analysis and exploratory signal interpretation. What began as a temperature-logging experiment evolved into a passive detection method for environmental electromagnetic disturbances. The fact that speech-like patterns appear reliably in specific seasons and during geomagnetic storm windows suggests that this is not merely coincidence or pareidolia amplified by compression.

While it’s impossible to definitively prove that these structures are directly caused by space weather, the weight of the alignment, seasonal trends, and resilience to audio filtering strongly supports that interpretation. The TEMPer1F thermometer—likely never designed with this use case in mind—appears to function as a crude EMF transducer under certain conditions. The presence of local electronics (e.g., powered speakers, networking gear, and a Wacom tablet) further complicates the environment, potentially modulating or amplifying ambient field activity.

There are, of course, limitations. The thermometer's resolution is relatively coarse, and its internal design is not known. Additionally, audio perception biases (such as a tendency to detect speech-like patterns in random stimuli) must always be kept in mind. Still, the persistence of these structures across multiple years and multiple software implementations builds a compelling case for continued investigation.

8. Conclusion

This project began with a quirky idea—turning temperature logs into audio—and ended with something that straddles the line between environmental sensing and unintended instrumentation. The consistency of speech-like patterns, their seasonal persistence, and their alignment with known solar and geomagnetic events all suggest that what's captured here is more than noise. Whether it's through direct electromagnetic interference, internal sensor behavior, or some mix of both, the TEMPer1F thermometer appears to be detecting and responding to real environmental fluctuations.

Through this process, both the method and the results have been validated: different codebases produced identical audio, patterns were robust to filtering, and timing aligned with real-world events. While this doesn't prove the data contains literal messages or conscious signals, it does raise serious questions about what kinds of environmental energy our everyday electronics may be passively picking up.

There's still plenty left to explore—better sensors, more refined analysis, and different types of data entirely—but this work lays a solid foundation for taking those next steps.

---

9. Future Work

Several directions remain open for deeper exploration, refinement, and validation of these findings:

- **Higher-Resolution Instrumentation**: Replace the TEMPer1F with a more precise and electromagnetically shielded temperature sensor to reduce noise and isolate true signal.
- **Dedicated EMF Logging**: Add a magnetometer or dedicated low-frequency EMF sensor nearby to cross-reference magnetic field fluctuations with the temperature-derived signal.
- **Sensor Shielding Tests**: Run the same setup inside a Faraday cage or with localized shielding to assess how much ambient interference contributes to the observed audio patterns.
- **Cross-Dataset Sonification**: Apply the same time-compression and sonification method to unrelated datasets (e.g., geomagnetic indices, tide data, financial time series) to compare pattern structure.
- **ASR and Phonetic Analysis**: Run structured speech recognition models on the audio to identify whether AI systems also detect phoneme-like features or interpretable fragments.
- **Interactive Visualizations**: Build a web-based interface that synchronizes waveform, spectrogram, and solar event markers for exploratory playback and analysis.
- **Broader Data Logging**: Expand the logging to include power line noise, CPU activity, or indoor EMF levels to rule out (or confirm) secondary influences.

10. Appendix

A. Spectrogram Snapshots

- Segment 1 (May–Nov 2018): Displays consistent harmonic banding in the 300–1200 Hz range.
- Segment 2 (Apr–Oct 2019): Similar structure with stronger dynamic transitions.
- October 2021 Pulse Event (Sample 499736): Brief, distinct vertical energy band indicating a sudden shift.

B. Solar and Geomagnetic Event Table

Event Type	Date Range	Sample Range	Description
G2 Storm	2018-05-05 to 05-06	251745–306937	Onset of structured audio
G3 Storm	2018-08-25 to 08-26	~290000–~295000	Strong mid-pattern structure
M-class Flares	2019-05-06	Within Segment 2	Enhanced upper-mid harmonics
G2 Storm	2019-08-31 to 09-01	Segment 2	Highly dynamic transition area
G1–G3 Storm	2021-10-03	499736	Audible “pulse”

C. Simulated Dataset Comparison Audio

- Audio derived from the fully modeled synthetic dataset.
- Compared against equivalent real-world segments.
- Demonstrates lack of mid-frequency harmonic clustering and absence of speech-like cadence.

D. Tools and Scripts

- PHP: `audioGen.php`, `wave.php`
- Python: data parser, audio converter, spectrogram generator (matplotlib-based)
- Filtering: SoX high-pass, bandpass, and noise profiling for Audacity

E. Code Listings

E.1 AudioGen PHP Class

Available at https://github.com/nitrocosmstudios/signals/blob/master/dat/classes/audioGen.php

E.2 Temperature Sonification in PHP

PHP code for converting a list of temperature samples into a 16-bit mono WAV file at 8 kHz playback rate.

```
require('./classes/audioGen.php');
$audio = new audioGen(8000);
$data = Array();
$dirname = '../archive/data/';
$dir = opendir($dirname);
$files = [];
while(($file = readdir($dir)) !== false){
  if(($file != '.') && ($file != '..')){
    $files[] = $file;
  }
}
sort($files);
foreach($files as $file){
  echo $file."\n";
  $fh = fopen($dirname.$file,'r');
  $file = fread($fh,filesize($dirname.$file));
  $file = explode("\n",$file);
  foreach($file as $line){
    $line = explode(' ',$line);
    if(isset($line[1])){
      $point = $line[1];
      $sample = intval(round(($point / 100) * 65536));
      $data[] = $point;
      $audio->addSamples(pack($audio->bf,$sample));
    }
  }
}
file_put_contents('./temper.wav',$audio→buildWAV());
```

E.3 Temperature Sonification in Python

Python code for converting a list of temperature samples into a 16-bit mono WAV file at 8 kHz playback rate.

```
import numpy as np
import wave
import struct
# Load your temperature samples (replace with actual loading logic)
# For example: temperatures = np.loadtxt("data/temps.txt")
temperatures = [...]  # List or NumPy array of float temperature values
# Step 1: Clip extreme values and scale to 16-bit range
min_temp = 40.0
max_temp = 110.0
temps_clipped = np.clip(temperatures, min_temp, max_temp)
scaled = ((temps_clipped - min_temp) / (max_temp - min_temp) * 65535 - 32768).astype(np.int16)
# Step 2: Write to a mono WAV file at 8000 Hz (one sample = one 5-minute reading)
sample_rate = 8000
output_path = "temperature_audio.wav"
with wave.open(output_path, 'w') as wav_file:
    wav_file.setnchannels(1)          # Mono
    wav_file.setsampwidth(2)          # 16-bit samples
    wav_file.setframerate(sample_rate)
    for sample in scaled:
        wav_file.writeframes(struct.pack('<h', sample))
```

F: Table of Figures

Figure	Title	Description
Fig. 1	Spectrogram – Snippet 1 (May–Nov 2018)	Structured harmonic bands; G2/G3 storm activity
Fig. 2	Spectrogram – Snippet 2 (Apr–Oct 2019)	Dynamic modulation; M-class flares and G2 storm
Fig. 3	Waveform – Snippet 3 (Oct 2021 Pulse)	Sharp vertical spike; aligned with G1–G3 storm
Fig. 4	Geomagnetic Sonification (2018)	Lacks structured harmonic features
Fig. 5	Geomagnetic Sonification (2019)	Broadband noise without speech-like structure
Fig. 6	Event Alignment Timeline (2018–2021)	Overlaid geomagnetic/solar events on temperature time series
Fig. 7	Waveform Comparison	A comparison between the outputs of the PHP and Python sonification scripts.

Figure 1: Spectrogram – Snippet 1 (May–Nov 2018)
Spectrogram showing structured harmonic bands and speech-like features aligned with G2/G3 geomagnetic storm activity.

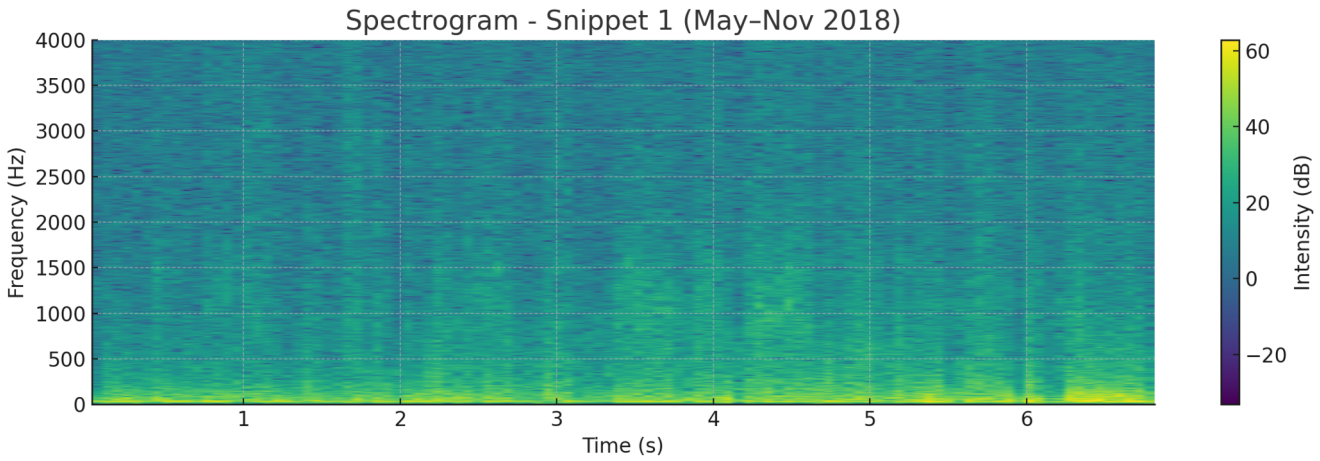

Figure 2: Spectrogram – Snippet 2 (Apr–Oct 2019)
Rich harmonic content and dynamic modulation, coinciding with M-class solar flares and a G2 storm.

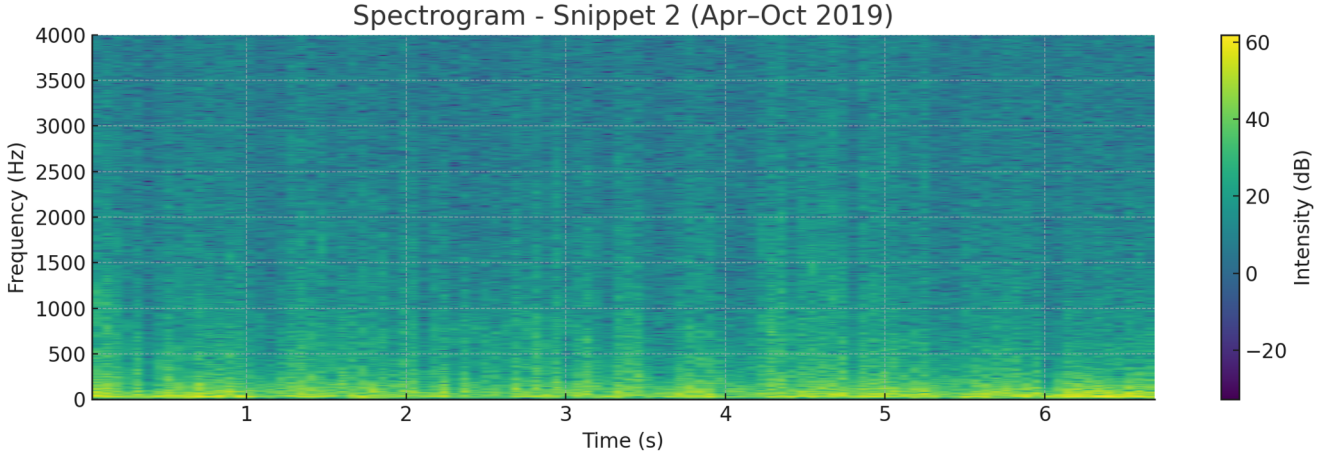

Figure 3: Waveform – Snippet 3 (Oct 2021 Pulse)
Sharp vertical energy spike at sample 499736, aligned with a G1–G3 storm window.

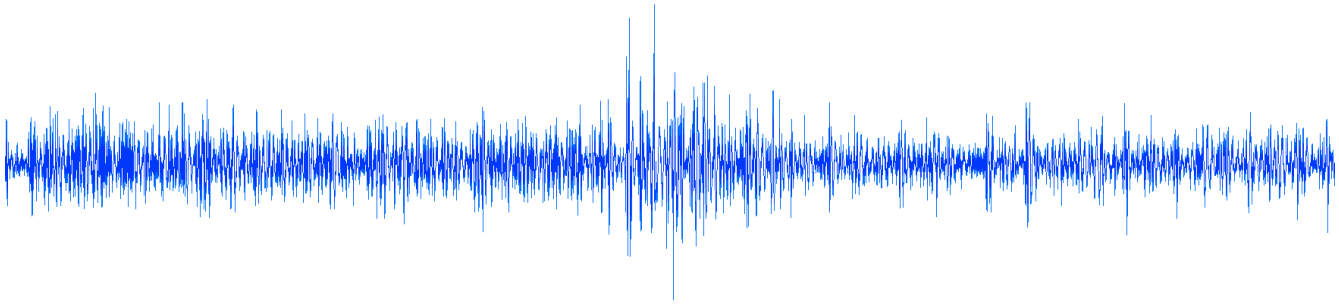

Figure 4: Spectrogram – Geomagnetic Sonification (2018)
Spectrogram from sonified Kp index data during Snippet 1 timeframe; lacks structured harmonic features.

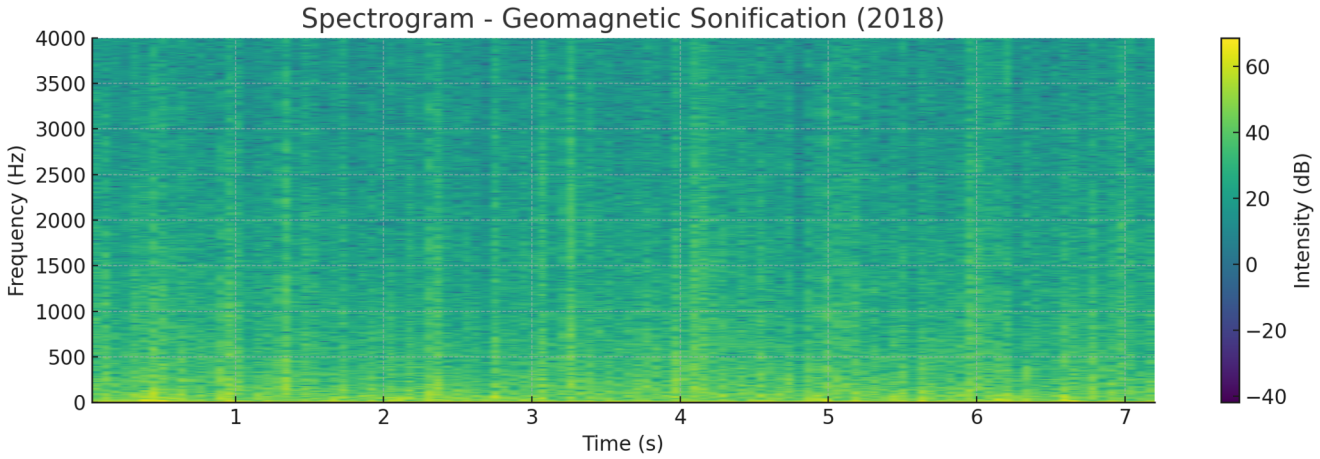

Figure 5: Spectrogram – Geomagnetic Sonification (2019)
Sonified geomagnetic activity aligned with Snippet 2; shows broadband noise without speech-like qualities.

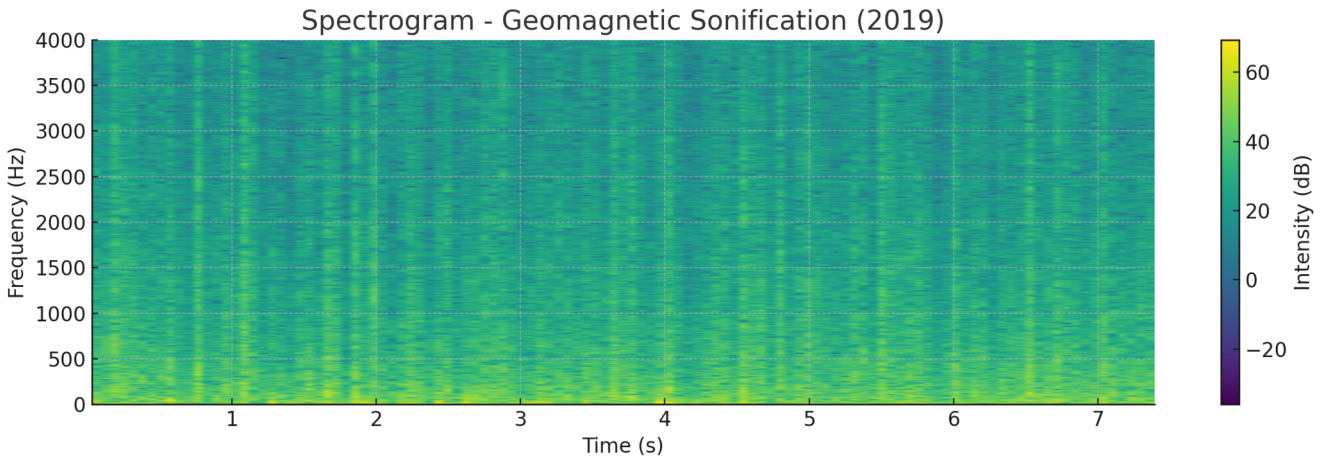

Figure 6: Event Alignment Timeline (2018–2021)
Timeline showing significant geomagnetic and solar events (highlighted with colored bands) overlaid on the raw temperature dataset. This visual correlates structured audio intervals with space weather activity.

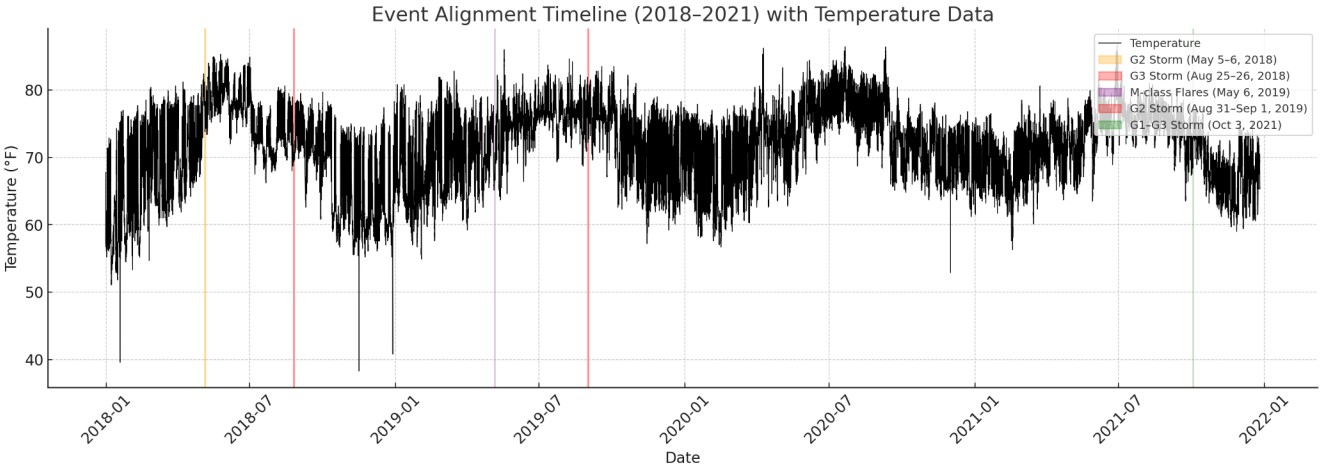

Figure 7: Audio Waveforms (2018–2021)

Waveform comparison of PHP and Python sonification output. The top waveform represents the PHP script's output. The bottom waveform represents the Python script's output. While not identical at the byte level, both exhibit the same structural features, cadence, and overall signal behavior.