The Ansible

A Space Data Synthesizer

Tate Fisher
BSE Electrical Engineering, ‘23
CoE Honors Capstone

ENGINEERING
HONORS PROGRAM
UNIVERSITY OF MICHIGAN

Abstract

The Ansible is a digital wavetable synthesizer that uses measurements of cosmic microwave background radiation (CMBR) collected from the Wilkinson Microwave Anisotropy Probe as the basis for its waveforms, allowing people to interact with the data in new and interesting ways.

Introduction

Sonification

Sonification is a method of data visualization where data is represented as audio. It can be accessible to individuals with impaired vision, and if done properly can assist in spotting patterns that would otherwise be hidden on a chart or graph: the EKG, for example, maps a patient's heartbeat to a "beep" that can help nurses and doctors detect irregularities without having to measure out peaks on a graph. Like other methods of data visualization, though, it's vulnerable to being overmanipulated, losing some critical aspects of its source. More specifically, there is a tendency with some sonification efforts to make it sound more attractive, more "musical", implying that the data itself is making this music, and not the individual who decided to map certain parameters of the data to things like pitch and timbre. As James Saunders notes in his essay *no mapping*, "When this mapping is presented as a kind of truth, suggesting we are listening to the sound of solar wind or climate change, this arbitrariness compromises both the data and the resulting music. [1]". This is all to say that, like with all data visualization efforts, it's important to be careful to not misrepresent the information by prioritizing how it looks or sounds over what it represents.

The main concern of this project isn't the legitimacy of certain types of sonification, mostly because it's just outside of the scope of this project - though it is important to keep it in mind when designing one. Instead, the focus is on changing how the audience *interacts* with it. Most sonification efforts are presented as a composition, a final sound that the audience listens to. There are plenty of examples of visual presentations that are interactive, so why isn't the same true for sonification?

The goal of this project, then, is to develop a possible method of sonification that is interactive and maintains a critical aspect of the data.

The CMBR

Space data is a popular source for sonification since it can be difficult to visualize things we normally can't see, like X-rays or infrared light emitted by distant bodies. This project uses space data as its foundation, but instead of focusing on a single source, the focus is on the universe itself – specifically the cosmic microwave background radiation, or CMBR. First measured in the 1960s, the CMBR is essentially the leftover heat from the Big Bang, radiated as light, with an equivalent temperature of roughly 2 degrees Kelvin (only two degrees above absolute zero!).

Initially, it was thought that it would be isotropic, measuring the same temperature no matter where in the sky it was measured. As instruments became more precise, it became more evident that the CMBR *does* vary depending on where in the universe it was being measured - making it anisotropic. This was the focus of the Wilkinson Microwave Anisotropy Probe (WMAP), launched by NASA in 2001 [2]. It measured the slight variation in temperature across the universe, and processing the data produced the image in Figure 1.

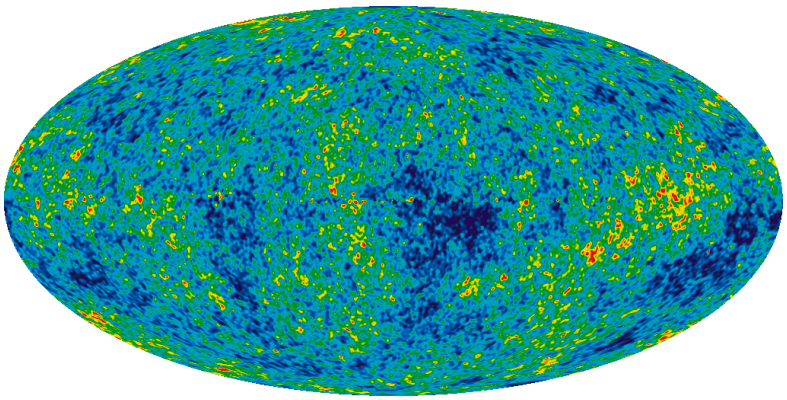

Figure 1: The cosmic microwave background radiation, or CMBR. This is the cumulated result of the 9-year measurements taken by WMAP. Image credit: NASA/WMAP Science Team.

What's interesting is that the variation in temperature is very, very small: only 2 μ, or 0.0002 K. Despite the temperature difference being this small, it's notable that there is a difference at all – which is why this project focuses on it.

Wavetable Synthesis

Since this project is intended to be interactive, it made sense to research methods of sound synthesis to determine the best way to make an interactive synthesizer out of data. As it turns out, one method is almost perfect for the representation this project aims for: wavetable synthesis. Essentially, instead of other methods where the basis is a single waveform that gets multiplied, added, and filtered (FM synthesis, for example), wavetables use a set of waveforms in computer memory to generate each period of the output. Each entry in the table is interpolated smoothly from each other, so that changing what wave is playing isn't abrasive. What this allows for is for a person using the synthesizer to "sweep" across many different arbitrary waveforms, resulting in a change of timbre not possible with other methods of synthesis. Why is this good for this project? If we can somehow convert the CMBR into a set of waveforms, we can put those waveforms into a wavetable synth and sweep through the data – allowing us to perceive the slight changes in temperature through the change in timbre as we change where we're measuring through our position in the wavetable.

Motivation

This project is my effort to make a sonification of the CMBR that conveys some aspects of that data accurately while also being interactive. The idea I eventually arrived at is to create a device that people can make music with, that also maintains what makes the CMBR interesting: its anisotropy. At its core, The Ansible is a wavetable synthesizer where each wave is generated from temperature data of the CMBR longitudinally. Its name, *The Ansible*, comes from the novels of Ursula K. Le Guin: a fictional device that allows for faster-than-light communication between anyone in the universe. I thought it would be fitting for a device that brings the user close to the measurable edge of it.

Methods

Processing the data

This portion was the focus of a significant part of my work on this project. I wanted to do the sonification well, and so much of my process was tweaking the process by which the data was converted to sound to preserve as much of the data as I could while also completing what I set out to do in the first place.

CMBR to Audio

The dataset used for The Ansible was the full-sky results from the WMAP's 5th year of study (Figure 2) for a few reasons. While it's not the most up-to-date version of the data (the most recent would be the 9th year results), the differences between the two sets are negligible for the purposes of this project. Additionally, and more importantly, it's the only dataset that also came in rectangular format, similar to a Mercator projection of a globe. This was important because it saved time that would have been spent converting the image provided by the WMAP team (see Fig.1). The raw data could have been used as well, but the image was used here for ease of access.

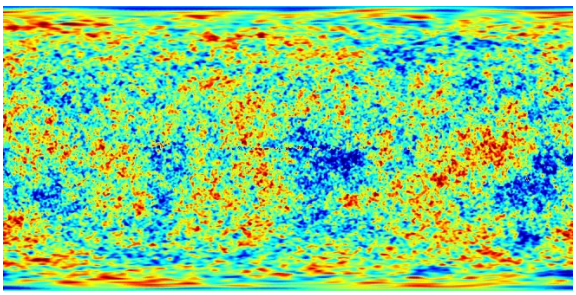

Figure 2: 5-year full sky data in rectangular format. Image credit NASA / WMAP Science Team.

The temperature recorded by the WMAP is indicated by hue, with the warmer portions of the CMBR represented by red, and the cooler portions represented by blue. Using MATLAB, the hue data can be extracted from the image, giving us a black-and-white representation of the data, where red (warm) is black and blue (cool) is white. Figure 3 displays the results of this process.

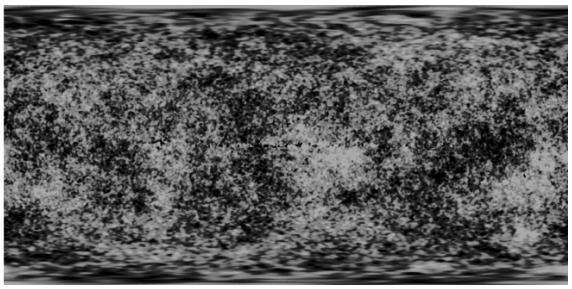

Figure 3: Hue data from the 5-year full sky data.

The next step from here is to extract waveforms from the hue data. First, the image was reduced to 256 columns and 512 rows for reasons that will be important later. Then, the waveform was generated by taking each column of the image and reading it as if it was an acoustic wave, reading down the column for each value. The hue data varied from 0 to 1, and so the resulting waves also varied from 0 to 1. This "unwrapping" of the image was performed by column and not by row because the first and last waveforms would be constant if rows were used.

The only processing of the data after this was removing the column's DC offset by subtracting its average value. This was done on a per-column basis and resulted in the wave varying from -0.5 at the lowest to 0.5 at the highest. Then, these waves were concatenated into one large waveform. This long waveform was converted to an audio file by assuming a sampling rate of 44.1 kHz (standard for audio) and exporting the long waveform as a .wav file. The concatenation was done for ease of storage, as one .wav file is easier to use than 256 tiny .wav files. The final audio file is about two seconds long, and very abrasive sounding.

A brief aside –this representation of the data is a little bizarre, since the waveform is "flipped" from what the data represent. The cooler portions are represented by a higher amplitude of the wave, and the warmer portions are represented by a lower amplitude. I didn't adjust this, since the adjusted waveform is just a flipped version of the original, which sounds identical to the original, unedited waveform.

Audio to Wavetable Synth

Now that we have our waveforms, we can implement the synthesizer. As mentioned earlier, we need to put these generated waveforms into a wavetable for the sonification

to be complete. Creating a wavetable synth from scratch would be a great exercise in audio programming, but would be reinventing the wheel given the plethora of audio tools available for free online.

SuperCollider (SC) is an open-source live audio framework that allows users to program their own synthesizers and audio effects. This project utilizes one key feature from SC, which is the SynthDef VOsc3: it's an arbitrary wavetable synthesizer that uses buffers containing the waveforms for its basis. The final hurdle for the synthesizer was then converting the audio file generated into this set of buffers. This is why the image was down sampled in the initial processing: by default, there are 1024 buffers available for use in SuperCollider, and so trimming the total buffers down to 256 (the number of waveforms generated) helps save space. Once this is done, the sonification is essentially complete.

The final flowchart from CMBR to Synthesizer is presented in Figure 4 below. This synth can be adjusted so that the position of the wavetable can be arbitrary or based on some input such as mouse position on the screen. Researching all of this - how to load the audio into the buffers and how VOsc3 works - ended up taking a fair amount of time to complete, as I had no prior knowledge of SuperCollider before beginning this project. At this stage, we can move on to the hardware implementation comfortably.

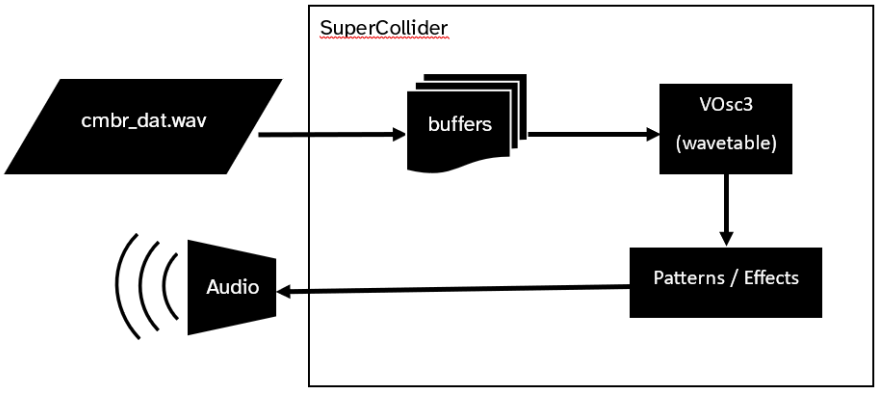

Figure 4: Flowchart for the CMBR audio file to wavetable synthesizer process.

Hardware Development

Development for the hardware happened in the opposite order of the digital synth: the block diagram (Figure 5) was created, and the implementation was based on it.

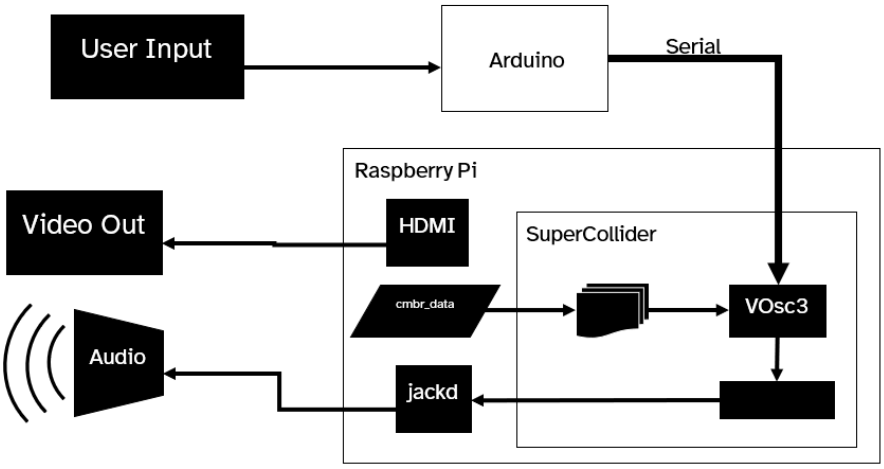

Figure 5: Hardware block diagram for the synthesizer.

The synthesizer created previously remains the same, but instead of running SuperCollider on a larger computer, it runs on a Raspberry Pi 4. Input is mapped from two knobs (a potentiometer and a rotary encoder) connected to an Arduino, which communicates with SuperCollider via a serial connection over USB.

SuperCollider on the Raspberry Pi

As it turns out, SuperCollider provides documentation for a Raspberry Pi build [3], for both GUI and GUI-less systems. In its initial draft I intended on having a fleshed-out GUI, and I find the desktop version of the Raspberry Pi easier to work with, and so I opted for the GUI version of SuperCollider. One issue I ran into when testing whether the installation worked was that the audio jack on the Raspberry Pi wasn't producing the correct output. This is a known issue with the Pi, and it was not recommended for use with SuperCollider [4]. Instead, I used an audio interface (Focusrite Scarlett Solo) for the audio output, which works well. The display is a small 5-inch touchscreen display, and I connected it to the Raspberry Pi's HDMI jack, as well as to several of its GPIO pins via DuPont cables.

Arduino to SuperCollider Connection

Initially, there was not going to be an Adruino. Instead, the sensors were going to be directly connected to the Raspberry Pi, and some code would read it in and pass the information to SuperCollider. Unfortunately, most of the GPIO pins I was going to use were occupied by the display, and so I instead opted to use an Arduino connected to the Raspberry Pi.

Sometimes a solution feels wrong, but it works well enough that you can ignore it. This is what developing the connection between the user input – a potentiometer and rotary encoder – to SuperCollider felt like. Developing the code for the Arduino was simple enough to do, as reading in data from sensors and buttons is one of the first things that someone writes for it. Communication was the tricky part – and the solution was interesting. Data from the sensors was sent over Serial (USB), and Supercollider can read from that data stream, which is great, except for the fact that three different pieces of information needed to be passed along, and the total digits sent weren't going to be the same each time. To separate these three streams, letters were inserted ("a", "b", "c") at the end of each piece of data. SuperCollider then detects when a letter was sent and stores the previously sent digits in an appropriate variable. This method is used by Eli Fieldsteel in his SuperCollider tutorials [5], which proved indispensable for this project.

For the Design Expo, the setup was that the potentiometer changed the position of the wavetable, and the rotary encoder changed the frequency that was playing. The encoder also had a built-in push button, and so I was able to have it change multiple properties, which for the expo became three different frequencies that it could be set at. In the future, this can be used to change any parameter, including cutoff values for filters, tempo for a sequencer, and other customizable parameters.

Case Design

The last step of the project was to design the case for The Ansible, which other than creating short pieces for it was one of the most creatively free parts of the design. In early sketches, I drafted two versions – one that resembled an old radio, and another that resembled an old CRT monitor, similar to what computers in the 1990's looked

like. The final draft is shown in Figure 6. It's similar to a microscope with the two knobs on the sides, which in my opinion gives it a more scientific feel. Either way, the tactile nature of it was important, and informed a lot of the hardware development. It's worthwhile to note that this draft had space for an internal speaker, which is missing in the final prototype due to the audio issues I had mentioned earlier.

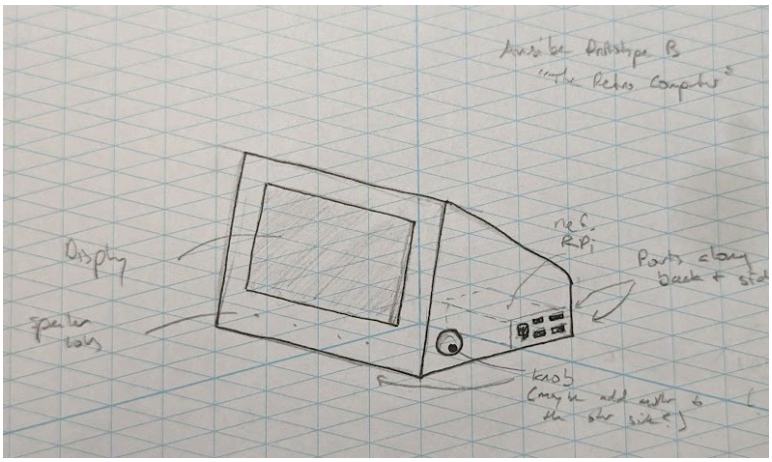

Figure 6: Initial draft of The Ansible's case.

The case was then prototyped with the screen out of cardboard (unfortunately I don't have any pictures), and the measurements of this protype were used for a 3D model created in Fusion 360. This model was then divided and 3D printed using black PLA filament. Figure 7 shows both the 3D model, and the final printed prototype.

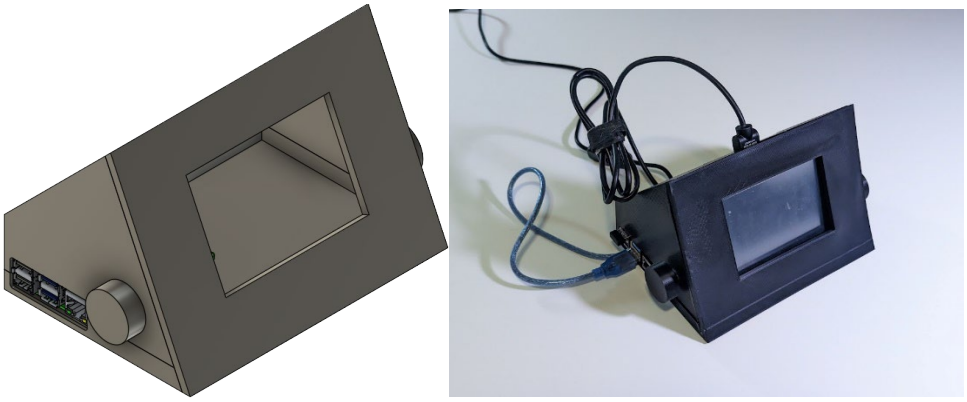

Figure 7: Final 3D model (left), and fully printed case with electrical components (right)

With this, the prototype for The Ansible was complete.

Discussion

Overall

The resulting device is, to my delight, actually fun to play with. Even just in SuperCollider, the sound of the wavetable moving across the data is a really engaging experience. The physical device feels like the right size, and with just the settings from the Design Expo it can make some very cool sounds.

I should bring up the Expo since that's where most of my concern was focused. I knew that I would be able to design something that I gained knowledge from and enjoyed using, but that doesn't mean that the point I wanted it to make was made. In hindsight, I should have had a poll that people could fill out before and after – but that could also be the scope creep talking. Judging by people's reactions, though, I think it's fair to assume that the device does what I set out for it to do.

Something I realized rather late in development is that this method could be used to generate a wavetable from any image, given you process it the same way. There is potential for use cases in image processing and the sonification of other sets of data, but I haven't been able to explore that avenue yet.

Future Work

Hardware

Rather than having to use an Arduino, it would be nice to have the two knobs connected directly to the Raspberry Pi, just to make the wiring much simpler. Additionally, changing the audio out from the interface to something more compact would greatly improve the device's look. Lastly, while the screen is a nice touch, I would like to experiment with what a GUI-less version of this would be like.

Software

The name of the game is the GUI. SuperCollider is a very powerful tool, but I found the GUI setup a bit limiting. Ideally, I would have the waveforms (or at least a small number of them) displayed on the screen, so that the user would be able to tell where

in the wavetable The Ansible is playing. That didn't end up working out, and I instead opted for SuperColliders' oscilloscope view to display the changing waveform.

Case

The case could use some more structural support, and could be better design to be easier to disassemble / reassemble.

Musical Composition

The works that I created to demonstrate The Ansible show the main working principle, but the goal would be to collaborate with other individuals in creating works using The Ansible as a basis, which is why the code is published in a public GitHub.

Conclusion

The Ansible conveys the anisotropy of the CMBR in a novel way, which allows for individuals to explore these data in an expressive manner. The addition of a physical device greatly improves the experience, but even in software The Ansible remains a unique method of sonification.

Works Cited

[1] J. Saunders, *no mapping*. MusikTexte, 2016. [Online] Available: http://musiktexte.de/WebRoot/Store22/Shops/dc91cfee-4fdc-41fe-82da-0c2b88528c1e/MediaGallery/Saunders.pdf. [Accessed: 23-Apr-2023].

[2] "Wilkinson Microwave Anisotropy Probe (WMAP)," *NASA*. [Online]. Available: https://wmap.gsfc.nasa.gov/. [Accessed: 23-Apr-2023].

[3]https://github.com/supercollider/supercollider/blob/develop/README_RASPBERRY_PI.md [Accessed: 23-Apr-2023].

[4] M. Skjeldgaard, "Notes for setting up a Raspberry Pi 4 for audio work". [Online]. Available: https://madskjeldgaard.dk/posts/raspi4-notes/. [Accessed: 23-Apr-2023]

[5] E. Fieldsteel. "SuperCollider Tutorial: 19. Arduino". August 23, 2017. Available: https://www.youtube.com/watch?v=_NpivsEva5o. [Accessed 23-Apr-2023].