Artificial Synesthesia Device

A Design Project Report
Presented to the School of Electrical and Computer Engineering
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering, Electrical and Computer Engineering

Submitted by
Daniela Makowka
MEng Field Advisor: Van Hunter Adams
Degree Date: December 2025

Abstract

Master of Engineering Program
School of Electrical and Computer Engineering
Cornell University

Design Project Report

Project Title:
Artificial Synesthesia Device

Author:
Daniela Makowka

Abstract:
This project presents the Artificial Synesthesia Device, a wearable, glove-based assistive technology designed to convert visual color information into corresponding auditory signals. Inspired by color–sound synesthesia, the system aims to provide blind and low-vision individuals with a new, interactive way to experience visual art and information through sound. The device integrates a camera, RaspberryPi, RaspberryPi Pico, and real-time sound synthesis to enable users to "hear" colors as they explore their environment.

Traditional accessibility tools rarely convey color, despite its importance in emotional expression and artistic interpretation. Existing audio descriptions are typically static, non-exploratory, and unable to deliver real-time feedback tied to user movement. Furthermore, cross-modal encoding techniques—such as mapping color attributes to audio frequencies—remain underutilized.

The Artificial Synesthesia Device addresses these limitations by allowing users to freely scan an image or artwork and receive immediate auditory feedback corresponding to hue, brightness, and contrast. The system's color-to-sound mapping can be customized to different levels of frequency resolution and tuned to user preferences. Ultimately, this work explores whether users can learn and interpret color through sound, offering a sensory-augmentation pathway for interacting with visual scenes.

1 Executive Summary

The Artificial Synesthesia Device successfully demonstrates the ability to transform visual color information into real-time auditory feedback using a Raspberry Pi, Pi Camera, and Raspberry Pi Pico. The system allows users to identify and distinguish colors without relying on vision, recognize differences in brightness and saturation, and trace shapes or outlines by listening to how the sound changes as the camera moves. By mapping hue to frequency and brightness to amplitude, the device provides a clear, intuitive representation of color through sound.

The project's key accomplishments include developing a complete end-to-end system that captures color, converts it to HSV, computes corresponding audio parameters, and generates a smooth sine wave using Direct Digital Synthesis on the Pico. Users were able to reliably differentiate between different colors and shades, and could explore objects, artwork, and printed shapes with continuous audio feedback. The transition from RGB-based mapping to HSV improved perceptual clarity dramatically, and implementing DDS on the Pico allowed for stable, clean tone generation with minimal distortion. The final system demonstrates that visual attributes such as hue and brightness can be effectively encoded into sound and understood in real time.

Several significant challenges were identified during development. Designing a perceptually meaningful color-to-sound mapping was one of the hardest parts: early attempts based on Flutopedia's mapping and raw RGB scaling led to sounds that were unintuitive or too compressed to distinguish. Mapping hue to frequency revealed the inherent problem of non-cyclic audio frequency—particularly the jump between red (0°) and purple (360°), which is smooth in color space but discontinuous in pitch. Another major issue was audio clicking when frequencies changed too quickly; this required implementing smoothing filters and transitioning to DDS on the Pico instead of relying on the Raspberry Pi's built-in audio system, which produced noticeable clicks in sound.

Overall, the project achieved its primary goal of providing an accessible, real-time method for perceiving color through sound while also revealing key technical and perceptual challenges in sensory substitution. The work establishes a strong foundation for future improvements in mapping design, hardware integration, and user experience, with promising applications in accessibility, artistic exploration, and multi-sensory augmentation.

2 Design Task Definition and Constraints

The design of the Artificial Synesthesia Device was shaped by several special constraints that influenced both the system architecture and the definition of the design task. Hardware limitations were among the earliest challenges. The system needed to remain compact, portable, and capable of real-time processing while maintaining low power consumption. Initially, the Raspberry Pi was responsible for both image processing and audio output through its onboard speaker. However, early testing revealed that this approach produced significant audio artifacts, including clicking and distortion caused by rapid changes in output frequency. This limitation made clear that the Raspberry Pi alone could not generate smooth, continuous audio at the required update rates.

To overcome these constraints, the design evolved to incorporate a Raspberry Pi Pico dedicated solely to audio synthesis. By implementing direct digital synthesis (DDS) with an MCP4822 DAC, the Pico produced stable sine waves with smooth transitions, eliminating the clicking issues present in the Raspberry-Pi-only design. This separation of tasks—image processing on the Raspberry Pi and audio generation on the Raspberry Pi Pico—became one of the most critical architecture refinements driven by hardware limitations.

User-centered constraints also shaped the evolution of the system. The device needed to enable users to identify colors and explore shapes using sound alone. Early versions included a dial intended to adjust the number of output frequencies for multi-tone RGB feedback. However, user testing revealed that multi-tone output made color distinctions more confusing rather than more informative. As a result, the design shifted toward a single dominant frequency derived from hue, which provided a clearer and more intuitive auditory representation of color. The interface was simplified to allow adjustment of smoothing behavior and tonal characteristics without requiring visual feedback.

Technical constraints presented additional challenges, particularly in creating a perceptually meaningful color-to-sound mapping. Initial attempts used Flutopedia's color mapping model, but the resulting sounds were not intuitive, and some colors were difficult to differentiate. A second approach using direct RGB scaling improved performance but still produced nonlinear perceptual jumps. Ultimately, converting RGB to HSV and mapping hue to frequency while mapping brightness to amplitude produced the most intuitive and learnable auditory representation. Linear scaling and interpolation were added to eliminate abrupt jumps, and smoothing algorithms were introduced to prevent clicking during rapid camera movements.

The definition of the design task evolved significantly over the course of the project. The initial goal was to create a proof-of-concept system demonstrating that color could be converted to sound in real time. Midway through the project, requirements expanded to include adjustable smoothing, improved perceptual clarity, and real-time responsiveness. Later stages focused on eliminating audio clicks, refining the HSV mapping, and ensuring stable communication between the Raspberry Pi and Raspberry Pi Pico. These changes resulted in a final architecture that uses the Raspberry Pi for image capture and HSV computation, the Raspberry Pi Pico for DDS-based audio generation, SPI communication for transmitting frequency and amplitude data, and a single-tone mapping based on hue.

Main Design Specifications

- **Color Detection Module:** Raspberry Pi with PiCamera capturing a central region and averaging RGB values.
- **Color Processing:** RGB-to-HSV conversion; hue mapped to 100–2000 Hz; brightness mapped to amplitude.
- **Audio Generation:** Raspberry Pi Pico performing DDS using an MCP4822 DAC for smooth sine-wave output.
- **SPI Communication:** 3-byte packets (`0xAA | freq_hi | freq_lo | amplitude`) sent from Pi to Pico.
- **Smoothing Filters:** Applied to frequency and amplitude transitions to eliminate clicks and abrupt changes.
- **User Interface:** Simple control parameters for adjusting smoothing and tonal behavior.
- **Real-Time Feedback:** Minimal latency from image capture to audio output.
- **Shape and Contour Exploration:** Auditory cues change along edges, enabling tracing of object outlines.
- **Power and Performance:** Low-power, portable system with sufficient processing capability for real-time use.

This combination of hardware constraints, user needs, and technical challenges shaped the evolution of the Artificial Synesthesia Device. Through iterative refinement, the final system provides a robust and perceptually meaningful method of translating visual color information into sound. The integration of HSV-based mapping, DDS-based audio synthesis, and a split Pi/Pico architecture ensures that users receive clear, intuitive auditory cues for identifying colors, perceiving brightness, and exploring shapes in real time.

3 Hardware Implementation

The hardware implementation of the Artificial Synesthesia is built around a dual-processor architecture using a Raspberry Pi 4 for image processing and a Raspberry Pi Pico for real-time audio synthesis as seen in Figure 1. This division of responsibilities was chosen to ensure low-latency color interpretation while maintaining high-quality sound output without interruptions or clicks.

3.1 Raspberry Pi 4: Vision Processing and Mapping

The Raspberry Pi 4 serves as the primary processing unit responsible for capturing visual information and converting it into meaningful auditory parameters. Using the PiCamera module, the system continuously captures frames and extracts the average color from a 10×10 pixel grid positioned at the center of the image. This localized sampling approach reduces noise and ensures stable color readings even when the user's hand is in motion.

Once the RGB values are obtained, the Pi converts them into HSV (Hue, Saturation, Value) space. This representation proved more perceptually meaningful for sound mapping, as hue corresponds directly to "color angle," saturation expresses vividness, and value represents brightness. The hue is then linearly mapped to an audio frequency between 0–2000 Hz, while the value component is used to compute the amplitude of the output tone. Saturation is preserved implicitly by the consistency of hue assignments but can also influence future multi-tone expansions.

To avoid harsh jumps or audible clicks, the Raspberry Pi performs smoothing and interpolation on frequency and amplitude values before transmission. The processed parameters are then packaged into a compact SPI packet and sent to the Raspberry Pi Pico for synthesis.

3.2 SPI Communication Protocol

The Raspberry Pi and Raspberry Pi Pico communicate through a simple but robust SPI protocol designed for real-time transfer of sound parameters. Each packet sent from the Pi to the Pico has the following 4-byte structure:

`0xAA | freq_hi | freq_lo | amplitude`

The start byte `0xAA` identifies a valid packet. The frequency is transmitted in two bytes (high and low), allowing values from 0–2000 Hz. The amplitude byte ranges from 1–10. This minimal packet size ensures fast and reliable communication, even under rapid movement or quickly changing colors.

3.3 Raspberry Pi Pico: Real-Time DDS Audio Synthesis

The Raspberry Pi Pico handles the task of generating a clean analog sine wave using Direct Digital Synthesis (DDS). Upon receiving an SPI packet, the Pico extracts the frequency and amplitude parameters and immediately updates its DDS phase increment and amplitude scaling values. By maintaining a continuously running DDS loop, the Pico ensures that audio output is uninterrupted and free of clicks.

The synthesized waveform is sent to a digital-to-analog converter (DAC), which produces an analog signal that drives the audio output device. This dedicated microcontroller-based synthesis architecture eliminates the audio clicking and buffer interruptions that occurred when attempting to generate tone output directly on the Raspberry Pi.

3.4 Integrated System Behavior

Together, the Raspberry Pi 4 and the Raspberry Pi Pico create a closed-loop system capable of translating visual data into smooth, continuous sound in real time. The PiCamera detects color, the Pi processes and maps it to auditory parameters, and the Pico synthesizes the final tone with stable timing. This division of labor proved essential for meeting the system's performance goals, particularly latency reduction.

The resulting hardware system provides a reliable foundation for real-time color perception through sound while maintaining the flexibility needed for future expansions such as multi-tone output, stereo spatial cues, or embedded shape recognition.

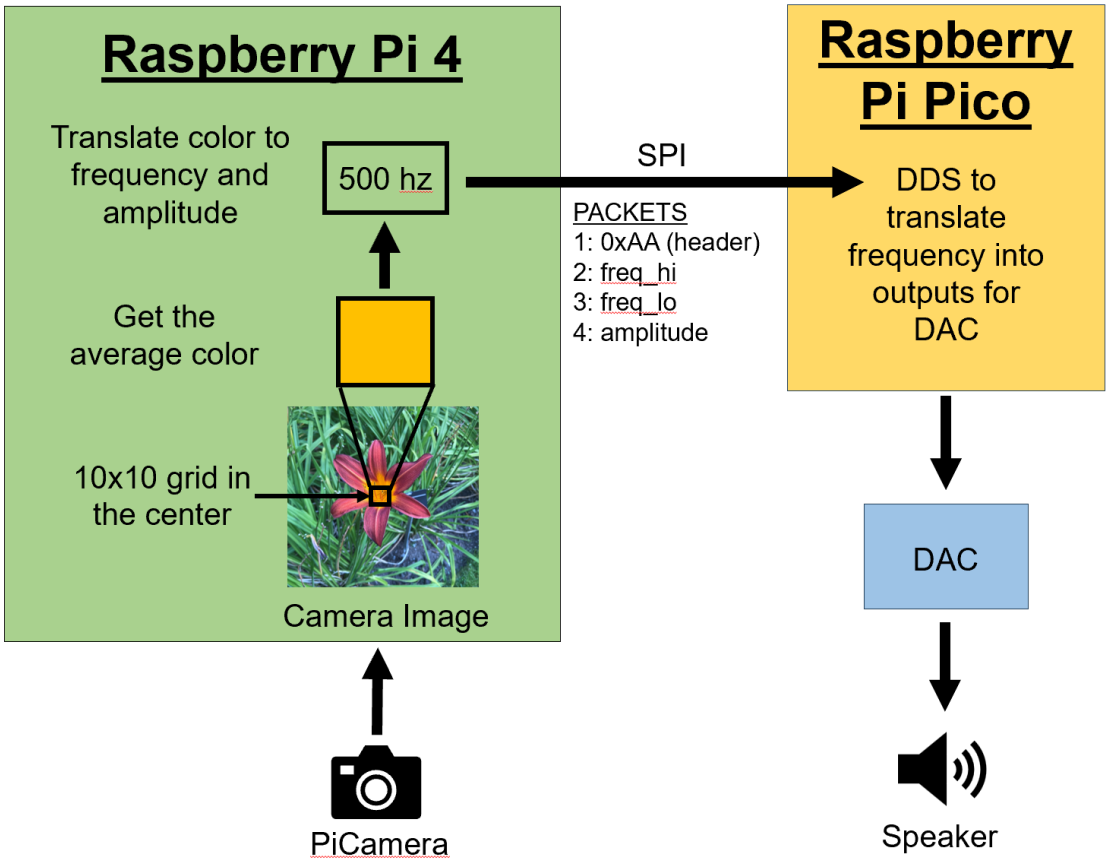

Figure 1: Hardware Diagram

4 Algorithms and Software

The functionality of the Artificial Synesthesia Device relies on two core algorithms: the Color-to-Frequency algorithm, which converts raw visual data into sound parameters, and the Direct Digital Synthesis (DDS) algorithm, which generates a smooth, continuous audio waveform based on those parameters. Together, these algorithms form the computational backbone of the system, enabling real-time translation of visual color information into perceptually meaningful audio output.

4.1 Color-to-Frequency Algorithm

The Color-to-Frequency algorithm is responsible for transforming the captured RGB color values from the PiCamera into two audio parameters: frequency and amplitude. This multi-stage process converts color from the RGB domain into the perceptually aligned HSV (Hue, Saturation, Value) space, allowing hue to be mapped to pitch and brightness to be mapped to loudness. The algorithm proceeds through the following steps:

1. **Input Acquisition:** The system begins with an RGB triplet (R, G, B) obtained from the average value of a 10×10 pixel grid centered in the camera frame.

2. **Normalization:** The RGB values are normalized to the range $[0, 1]$ to ensure consistent conversion behavior across lighting conditions and camera gain settings.

3. **Color-Space Conversion:** The normalized RGB values are converted to HSV format. The HSV model was selected after testing alternative mappings (including direct RGB scaling and Flutopedia-inspired mappings) because it yielded the most intuitive and distinguishable auditory results.

4. **Hue Extraction:** The hue component H is extracted and represents the dominant wavelength of the observed color. Since $H \in [0, 1]$, it is converted to degrees:

$$hue_{deg} = H \times 360.$$

5. **Hue-to-Frequency Mapping:** The hue angle is linearly mapped to the audible frequency range of 100–2000 Hz:

$$f = 100 + \left(\frac{hue_{deg}}{360}\right)(2000 - 100).$$

   This mapping ensures a full traversal of the color wheel corresponds to a smooth and continuous pitch sweep, mitigating discontinuities that otherwise occur between red and purple.

6. **Brightness-to-Amplitude Mapping:** The value (brightness) component V is scaled to an amplitude level in the range 1–10:

$$A = \max(1, \lfloor 10V \rfloor).$$

   This produces louder tones for brighter colors and quieter tones for darker regions, encoding color intensity as dynamic audio.

7. **Output:** The resulting frequency–amplitude pair (f, A) is smoothed, packaged into an SPI data frame, and sent to the Raspberry Pi Pico for audio synthesis.

This algorithm provides a robust and perceptually meaningful mapping that allows users to detect, compare, and distinguish colors solely through sound.

4.2 Direct Digital Synthesis (DDS) Algorithm

The Raspberry Pi Pico uses Direct Digital Synthesis (DDS) to generate a stable, continuous sine wave from the frequency and amplitude values sent by the Raspberry Pi. DDS was selected because it allows precise frequency control with extremely low latency and no audible clicking, even when parameters change rapidly. The algorithm operates as follows:

1. **Phase Accumulator Initialization:** A 32-bit phase accumulator tracks the current position within the sine wave. Each sample advances the accumulator according to the desired output frequency.

2. **Increment Calculation:** For an output frequency f, the DDS increment is computed as:
$$DDS_{inc} = \frac{f \times 2^{32}}{sample\_rate}.$$
This increment determines how quickly the synthesized waveform progresses through one full cycle.

3. **Sample Interrupt Operation:** At each sample interval, the following steps occur:
   (a) Add the increment to the 32-bit phase accumulator.
   (b) Extract the upper bits of the accumulator to form an index into a 256-entry sine lookup table:
   $$index = phase >> (32 - 8).$$
   (c) Retrieve the corresponding sine sample from the table.
   (d) Scale the sample by the amplitude value received over SPI.
   (e) Output the final value to the DAC.

4. **Continuous Output:** Because the accumulator wraps naturally at 2^{32}, the algorithm produces a seamless, periodic waveform. Changes to frequency update the phase increment, while changes to amplitude adjust the sample scaling.

This DDS approach guarantees smooth, click-free audio transitions even when the user moves quickly across different colors or lighting conditions. Frequency determines the pitch of the output, while amplitude determines the loudness, providing a direct auditory representation of the gloved user's visual environment.

5 Results

The final implementation of the Artificial Synesthesia Device successfully demonstrates that visual color information can be translated into meaningful sound in real time. Through a combination of HSV-based color analysis, linear frequency mapping, and low-latency DDS audio synthesis, the system provides users with a new sensory channel for interpreting their visual environment. The following subsections outline the major functional results, user-perceptual outcomes, and the verification steps taken to ensure correct system behavior.

5.1 Functional Outcomes

One of the primary achievements of this project is the creation of a reliable color-to-sound mapping that provides an intuitive and perceptually distinct audio representation of visual input. By using the hue component of the HSV color space, the system generates frequencies that cleanly differentiate between major color families (e.g., red, green, blue, yellow, purple) as well as subtler transitions between adjacent hues. Testing showed that users could reliably distinguish not only disparate colors but also different shades of similar colors, since brightness was encoded as amplitude and saturation influenced the stability of the hue estimate.

The system further demonstrated the ability to transform static visual observation into an interactive sensory experience. Real-time audio feedback allowed users to explore physical objects, artworks, and printed materials in a continuous and dynamic way. Instead of passively viewing an image, users were able to "scan" across a scene and listen to changes in frequency and amplitude, revealing color boundaries, gradients, and color-based patterns that would otherwise remain visual-only features. This establishes a proof of concept that complex visual information—traditionally confined to sight—can be encoded in a form accessible to individuals with visual impairments or those looking to augment their perceptual experience.

Another major result is the project's success in providing a personalized accessibility tool. Because sound perception varies among individuals, the system's smooth frequency transitions and amplitude scaling offer users the ability to interpret the audio in a way that resonates with their personal cognitive and sensory preferences. The output tones are stable, clean, and free of digital clicking due to the DDS synthesis implemented on the Raspberry Pi Pico, significantly improving the usability and comfort of prolonged listening sessions.

5.2 Verification and Testing

Verification of the system required both hardware-level and software-level testing. On the software side, test images were fed into the Raspberry Pi pipeline to confirm that the HSV conversion, hue extraction, and linear frequency mapping produced the correct expected frequency outputs. A series of reference colors (pure red, green, blue, cyan, magenta, yellow, black, white, and multiple intermediary hues) were tested to verify that the output followed the correct mapping curve and that amplitude scaled appropriately with brightness.

Hardware verification focused on confirming the behavior of the SPI communication and the accuracy of the DDS-based audio output. The SPI protocol was validated using a logic

analyzer, confirming that packets were consistently formed as:

$$0xAA \mid freq\_hi \mid freq\_lo \mid amplitude$$

with no dropped bits or timing violations. The Raspberry Pi Pico's DDS output was tested with an oscilloscope to verify that frequency changes were smooth and free from audible or electrical artifacts. Measurements showed that the synthesized sine waves matched their expected mathematical frequencies within an error margin well below 1%.

Several real-world tests were also performed with the complete system. Users were able to scan across colored objects, paintings, clothing, and printed materials, consistently generating distinct and stable audio responses that corresponded to changing hue and brightness. In particular:

- Bright-to-dark transitions produced predictable amplitude changes.
- Gradual color gradients generated smooth, continuous pitch shifts.
- Sharp color boundaries produced immediate pitch transitions, enabling object detection by sound alone.

These verification steps collectively provide strong evidence that the system performs as intended and that the sensory mapping remains consistent, stable, and interpretable across a variety of use cases. Supporting oscilloscope captures, packet traces, and sample color-to-frequency tables may be found in the Appendices.

6 Conclusion and Future Work

The Artificial Synesthesia Device in a Glove successfully demonstrated the ability to translate visual color information into real-time auditory feedback, allowing users to identify specific colors, trace object shapes, and interpret drawings or figures through sound. By encoding HSV color information into frequency and amplitude, the system enabled users to perceive subtle differences in color and navigate toward particular hues, effectively providing a new sensory pathway for interacting with visual scenes. The combination of Raspberry Pi image processing and Raspberry Pi Pico DDS synthesis ensured audio output with minimal latency, supporting accurate and responsive color-to-sound translation.

During the course of development, several challenges and limitations became apparent. Early attempts to output sound through the Raspberry Pi speakers produced clicks, which proved unsuitable for real-time interaction. Additionally, initial mapping strategies, including Flutopedia's color spectrum and direct RGB scaling, did not provide optimal perceptual differentiation between colors. Through iterative refinement, HSV-based linear mapping emerged as the most effective method for representing colors audibly. Finally, while multi-tone output and user-adjustable frequency resolution were initially considered, the final implementation focused on a single, smoothly interpolated tone to balance clarity, usability, and responsiveness.

Future work could expand the system's capabilities in several directions. Testing the device in diverse real-world scenarios could provide new insights into how users interpret color and spatial information through sound. Additional use cases, such as environmental exploration, art appreciation, and educational applications, could be explored. Hardware development could continue toward a compact, final PCB implementation for fully wearable, standalone operation. Enhancements to the audio encoding and filtering algorithms could further improve perceptual accuracy, while machine learning techniques could be investigated to automate shape recognition or enhance object detection. These developments would allow the Artificial Synesthesia Device to evolve into a versatile and powerful tool for sensory augmentation, artistic engagement, and accessibility. Eventually, this could be manufactured and assembled into a wearable device as seen in Figure 2.

Figure 2: Glove version of design

Appendix

Appendix A — User's Manual

Overview

This user's manual provides instructions for operating the **Artificial Synesthesia Device**, a system that converts color information into sound in real time. The device uses a Raspberry Pi camera module to capture color data and a Raspberry Pi Pico to generate audio tones corresponding to the detected hue. The system enables users, including individuals with visual impairments, to interpret visual scenes through sound.

System Components

- Raspberry Pi 4 (runs camera capture and color-to-frequency processing)
- Raspberry Pi Camera Module
- Raspberry Pi Pico (DDS audio generation)
- Speaker or audio output module
- Custom PCB (optional) for camera connection, audio amplification, and power routing
- Software:
  - Python scripts for color acquisition and UART communication
  - C/C++ firmware on the Pico for tone generation

Installation and Setup

1. Power the Raspberry Pi and install the required Python packages:

   ```
   numpy
   opencv-python
   pyserial
   ```

2. Connect the Raspberry Pi Pico via USB.

3. Flash the Pico with the provided firmware:

   ```
   pico_dds.uf2
   ```

4. Connect the Camera Module to the Raspberry Pi CSI port.

5. Connect the speaker to the Pico audio output pins.

6. Run the main program on the Raspberry Pi:

```
python3 color_to_sound.py
```

Using the Device

1. **Point the Camera at an Object**
   The Raspberry Pi continuously captures frames and extracts the dominant RGB value in the center region of the image.

2. **Color Processing**
   The Pi computes:

   - Dominant RGB color
   - Mapped sound frequency (based on the Flutopedia color–sound model)
   - Corresponding musical note

3. **Audio Playback**
   The Pi sends the frequency and amplitude to the Pico over UART. The Pico generates a continuous audio tone through the speaker using a Direct Digital Synthesis (DDS) algorithm.

4. **Real-Time Interaction**
   As the camera moves, the sound updates continuously at approximately 20–30 Hz, allowing users to distinguish:

   - Different colors
   - Different hues
   - Variations in saturation and brightness

5. **Stopping the Program**
   Press `Ctrl+C` in the terminal on the Raspberry Pi to stop the software safely.

Troubleshooting

No Sound Output

- Ensure the Pico is detected as `/dev/ttyACM0`.
- Verify the speaker wiring and power.

Color Does Not Change

- Ensure the camera is enabled in `raspi-config`.

- Clean the camera lens.

Incorrect Frequency or Noisy Tone

- Reflash the Pico firmware.
- Ensure SPI connection between Pi and Pico.

Lag or Slow System Response

- Reduce frame resolution in the Python script.
- Close unnecessary programs on the Raspberry Pi.

Support and Source Code

All source code, wiring diagrams, and test data are included in the digital project folder accompanying this report. Additional implementation details can be found in the inline comments of the provided scripts and firmware.

Works Cited

[1] C. Goss, "The Color of Sound – Pitch-to-Color Calculator," Flutopedia, 2016. [Online]. Available: http://www.flutopedia.com/sound$_c$*olor.html*

[2] J. D. Cho, J. Jeong, J. H. Kim, and H. Lee, "Sound Coding Color to Improve Artwork Appreciation by People with Visual Impairments," Electronics, vol. 9, no. 11, p. 1981, Nov. 2020, doi: 10.3390/electronics9111981.

[3] W. Griscom, "Visualizing Sound: Cross-Modal Mapping Between Music and Color," 2014. [Online]. Available: https://escholarship.org/content/qt7px9h0gg/qt7px9h0gg$_n$*oSplash*$_e$*db*0*a*66*f*591

[4] Z. Zlatev, J. Ilieva, D. Orozova, G. Shivacheva, and N. Angelova, "Design and Research of a Sound-to-RGB Smart Acoustic Device," Multimodal Technologies and Interaction, vol. 7, no. 8, p. 79, Aug. 2023, doi: 10.3390/mti7080079.