Whistle Lamp

The Whistle Lamp is a multifunctional lighting appliance designed to enable a hands-free user experience. The device utilizes a Fast Fourier Transform algorithm run on an ESP32 microcontroller to process audio input from an I2S microphone, adjusting light brightness or angle as output by sending a pulse-width modulation (PWM) signal to an LED and servo motor, respectively.

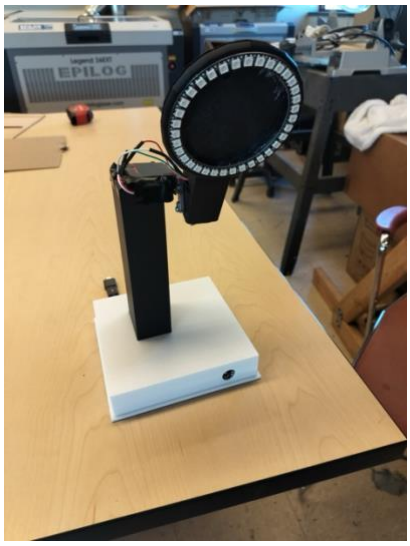


Fig 1: Final physical artifact

Components

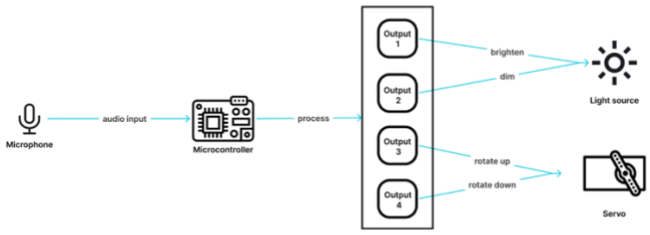


Fig 2: High level design

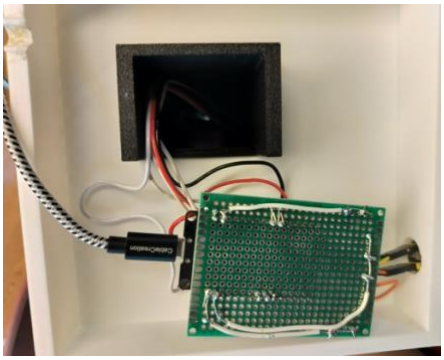
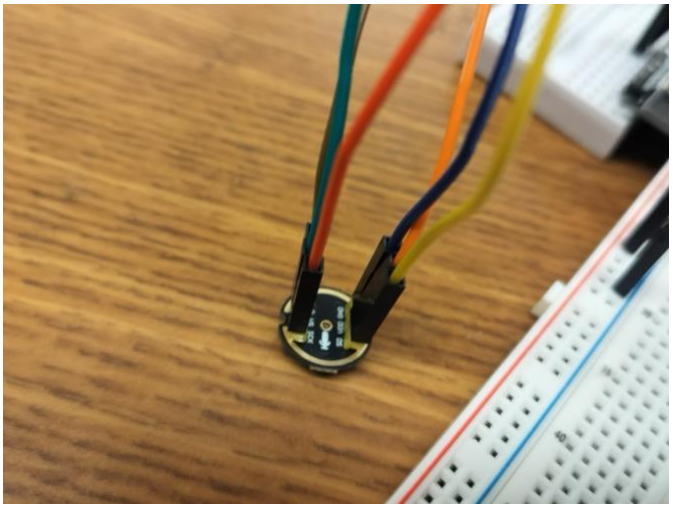
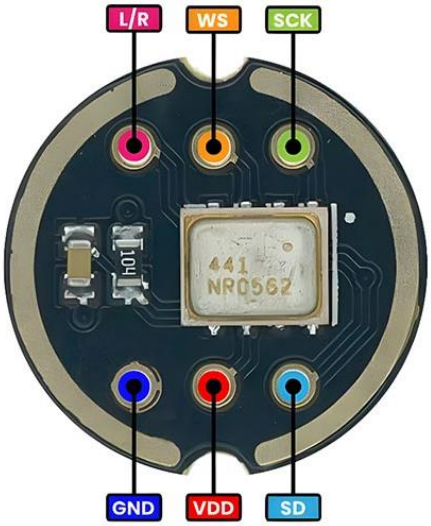


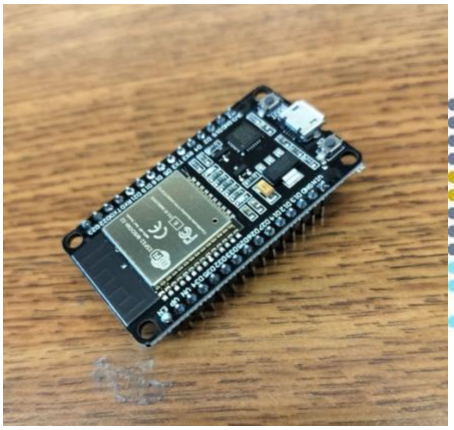
Fig 3: Soldered protoboard

- Microphone (I2S Interface INMP441 MEMS)
  https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf

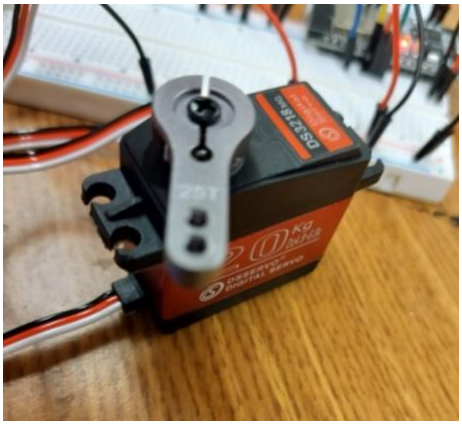




- Microcontroller (ESP32-WROOM-32)
  https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf



- Servo (DS3218MG 20kg Metal Gear 270°)
  https://images-na.ssl-images-amazon.com/images/I/81Lbgu+nG6L.pdf



- LED (WS2812 35-bit RGB)
  https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

Libraries

- FastLED https://fastled.io/
  - Arduino library for programming addressable LEDs

- AudioTools https://github.com/pschatzmann/arduino-audio-tools
  - Audio processing library for Arduino/ESP32

Parts

Fig 4: CAD rendering of physical artifact

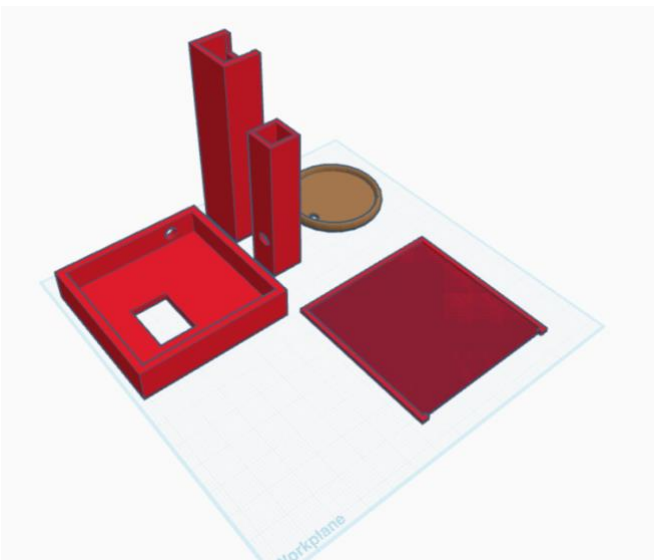


Fig 5: CAD rendering of individual parts

Skills

FTT – https://www.elektormagazine.com/articles/fast-fourier-transform-fft-on-the-esp32

- I learned the basic workings behind the Discrete Fourier Transform and Fast Fourier Transform including that a FFT is more efficient by decomposing the input frequency vector such that the DFT matrix can also be decomposed into multiple sparse matrices that allow for faster multiplications/processing, thus reducing time from n^2 to nlogn. I also learned that there existed many libraries that implemented the FFT algorithm much more efficiently than I ever could, leading me to follow the tutorial linked above to implement a basic tone detection program. In addition to the tutorial materials, I also designed a simple bandpass filter to eliminate some background noise as well as account for the harmonic tones of a whistle.

I2S

- I learned some I2S communication basics, as well as how to use it to transmit digital audio data to the microcontroller.

ESP32

- Learned how to program an ESP32 microcontroller, as well as utilize its peripherals.

Process

Initial testing

- Included testing servo and LED by reading documentation to find PWM signal ranges and running each component individually with a potentiometer as input.
- Audio testing involved attempting to detect a digitally generated sine wave tone via microphone and converting to frequency with microcontroller.

Ideas that didn't pan out

- Initial plans for user input did not follow through. I had wanted to program it with a binary input in mind, such that a tone below a threshold would represent a 0, above would represent a 1, so that the user could whistle different patterns to result in 00, 01, 10, 11 for the four respective outputs, however the FFT speed was not quite fast enough to implement this smoothly. Another possibility was to use a convolution neural network as suggested by Prof. Gilliland to notice rising/lowering tones, however a lack of coursework/knowledge in the area and not quite enough time led me to scrap that idea as well.

Lessons learned

- Be extra careful when designing 3D models, be precise with measurements and part shapes. Limit electrical noise as much as possible, solder everything if necessary.

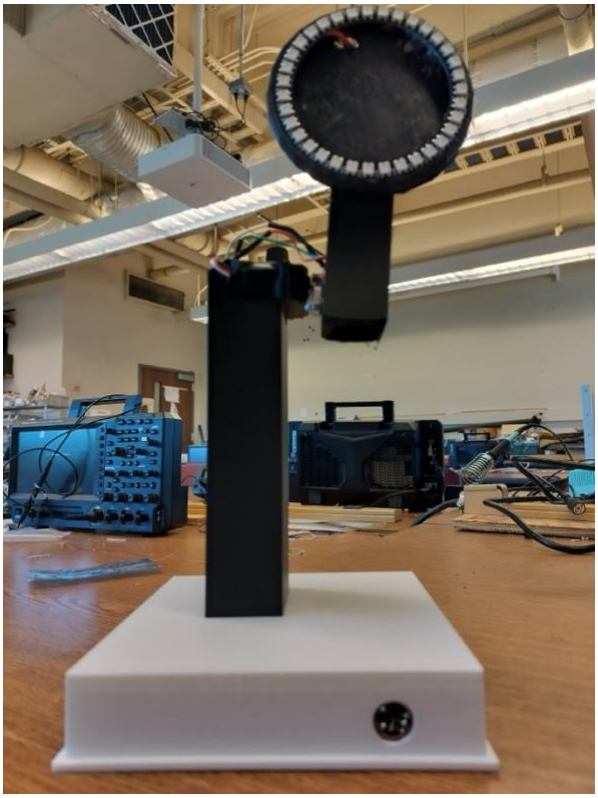


Fig 6: Physical artifact alternate angle view

Fig 7: Physical artifact back view