Multi-Frequency Based Controller

A. James Clark School of Engineering
University of Maryland

Department of Electrical & Computer Engineering

Team Members: Shaurya Agarwal, Nathan Fireman, Owen Mank, Rajit Mukhopadhyay, Jamil Takieddine
Professor: Dr. Brian Beaudoin | ENEE 408J

INSPIRATION

Bringing **Yondu's arrow** from **Guardians of the Galaxy** to life

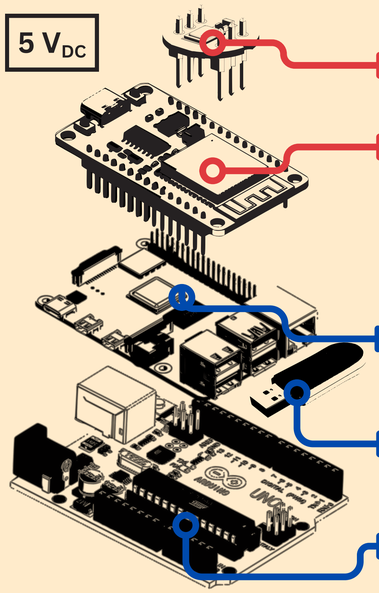
OBJECTIVE

A **configurable system** to **translate complex audio cues** into **independent output instructions** in the audible range for **sensing and control**

APPLICATIONS

- **Smart Homes (Consumer Tech):** Tone-based triggers to control lights, appliances or locks
- **Factories (Industrial Automation):** Systems can self-disengage if frequencies associated with mechanical failure are detected
- **Interactive Exhibits (Arts & Education)**: Allows visitors or students to trigger displays using musical notes

ARROW DESIGN



Transmitter

- INMP441 Microphone Preamplifier
- ESP32 Dev Module

Receiver

- Raspberry Pi 4 Microcontroller
- USB Configuration Device
- Arduino Controlled Device

Signal to Movement

- ESP32 Receives Raw Signal Data from Preamp
- Data Sent to Raspberry Pi Via Serial Bluetooth Connection
- Pi Runs Python Signal Processing Scripts on Raw Stream of Data
- Translated data is converted to control instructions
- Raspberry Pi passes output stream via Serial Connection to Arduino
- Arduino Processes Outputs for Programmed Action

SIGNAL PROCESSING

1. **Convert Signal** from I^2S to **dBSPL**
2. **Bandpass Filter** to audible spectrum
3. **Subtract Noise Floor** to **improve signal clarity**
4. **FFT** to isolate signal frequency peaks
5. **Peak Grouping** to **reduce processing complexity**

CONFIGURATION

JSON Format:

GLOBAL VARIABLES

- Configuration Name
- Signal Sensitivity (Hz)
- Amplitude Sensitivity (dB)

SIGNAL - OUTPUT MAP

- Frequency Peak (Hz)
- Frequency Sensitivity (Hz)
- Amplitude Sensitivity (dB)
- Output Code (String/Char)

EXAMPLE:

```
{
 "config_name":
"two_tone_test",
 "Global_Sensitivity": 10.0,
 "Global_Amplitude": 65.0,
 "Peaks": {
  "Peak1": {
   "frequency": 440.0,
   "sensitivity": 5.0,
   "output_code": "A"
  },
  "Peak2": {
   "frequency": 523.25,
   "sensitivity": 5.0,
   "output_code": "C"
  }}}
```

DEMO

Watch here:

... or try it yourself!

ANALYSIS

- **Electrical Noise Reduction:** Reducing the length of wire used and soldering our connections minimized electrical noise.
- **Constructive and Destructive Interference:** In its current state, signal clarity is impacteed when notes are played simultaneously.
- **Note Testing and Identification:** Once frequency detection was consistent, we recorded the frequency ranges generated by striking varying notes on a tongue drum. After this we identified 4 notes with fairly unique frequencies.
- **Testing Applications of Sensor:** Connecting our Raspberry Pi to a Lafvin Arduino bot, we programed it to drive forward or backward and to turn left or right based on which of the 4 notes it received.

TEAM YONDU