Check Point Summaries

• Week 2-3: Mandatory initial discussion with instructor about project ideas [2%]

• Document and summarize meeting via a direct non-anonymous message on Piazza to Instructor

Week 2-3 Initial Meeting Summary:

We had some project ideas that we had not fleshed out in detail and seemed too complex to achieve for the scope of this class project. We discussed these ideas with the professor and the issues that would have arisen. Professor Srivastava gave us a project idea that builds upon a previous year's project on Home Appliance Detection. The previous year's project was able to detect sounds one at a time however, we aim to expand on this by detecting multiple sounds at a time. We liked this project idea and will continue to do some research to build more understanding on how to achieve this goal.

• Week 3-4: Mandatory second discussion with instructor about project ideas 3%]

• Document and summarize meeting via a direct non-anonymous message on Piazza to Instructor

Week 3-4-5 Second/Third Meeting Summary:

We created the following document to flesh out our project details: https://docs.google.com/document/d/10HaTjYyHpSiAY9tTNf4n954gSxCeEPFE5EudunRQ_aE/edit?usp=sharing.

For our second meeting, Nathan had done research on sound classification models and sound source localization methods. Bhanu and Disha received feedback to contribute more and research specifics such as which hardware elements (microcontroller, microphone system) to use, software platforms such as machine learning tools and programming languages, and a high-level overview on the interconnections of components in our embedded system.

For our third meeting, we presented our findings on Edge Impulse and a specific sound classification model we had found. The model we found may have been too large for our purposes so we were advised to look for an alternative or use the model we found as an example and possibly prune it for our use. We also presented our choices on the hardware we will be using. We decided to use a Raspberry Pi 4 B as our microcontroller paired with a Google Coral Accelerator as well as the Respeaker Microphone Array V2. We discussed the research that was done behind making these choices. All of our findings are presented in the linked document. We were also able to solidify our goals and initial aims for this project.

• Week 5: Initial project repo and website ready on GitHub [5%]

• Submit project title, team information, GitHub repo URL, and GitHub website URL via web form

• Website must have initial sections for goals, specific aims,

Project Title: Home Appliance Detection

Team Information: Bhanu Chaudhary, Nathan Portillo, Disha Zambani
Github Repo URL: https://github.com/NatePortillo/esysproj
GitHub Website URL: https://dzambani.github.io/

==========

Methods/Research

Research Paper : Security Monitoring Using Microphone Arrays and Audio Classification *:*
https://ieeexplore.ieee.org/abstract/document/1658350?casa_token=JXTdeAo9oN4AAAAA:-UGeft3umHvMDsc6f6mzKfgAiUVWbGIdavgmeLObT2AmFvDQjpYze-Xutu6vbvG0TqgekK28

Research Paper : Pitch-based feature extraction for audio classification:
https://ieeexplore.ieee.org/document/1244723

Same resource. Different wording.

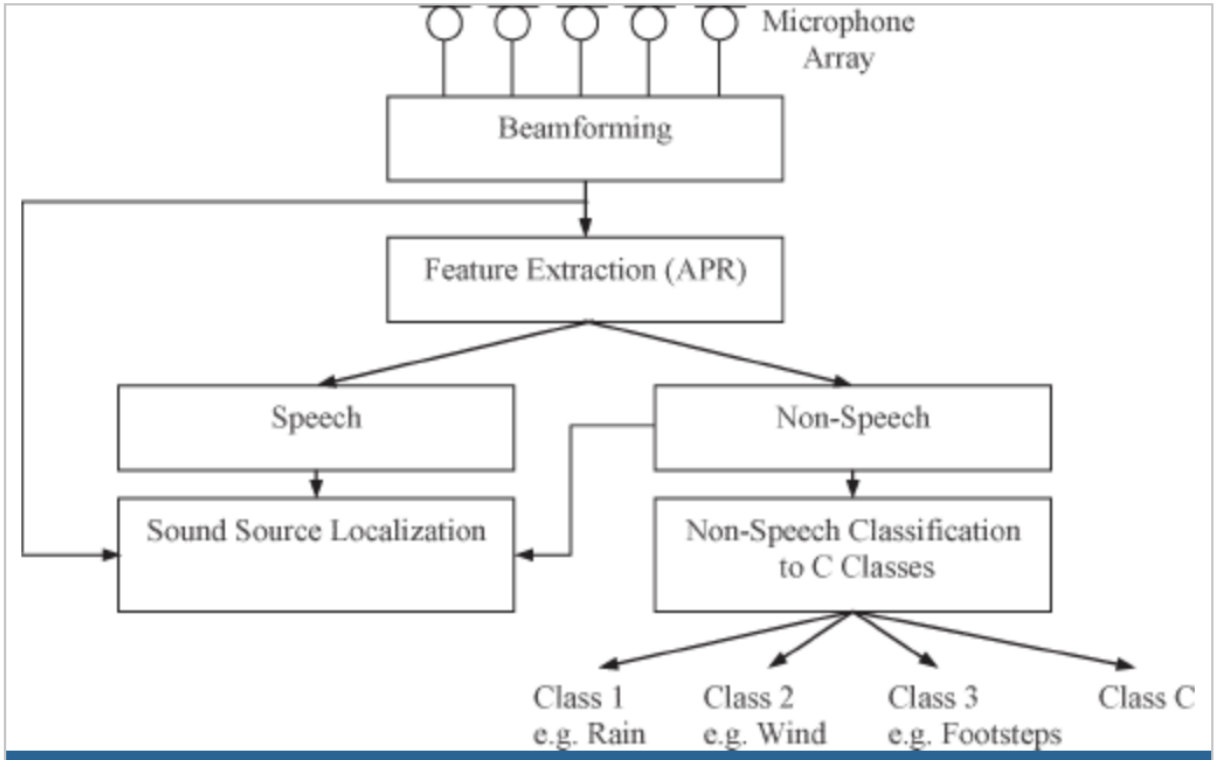
Abstract:

The project can detect and classify the location and nature of different sounds within a room. Works well in low signal-to-noise environments.

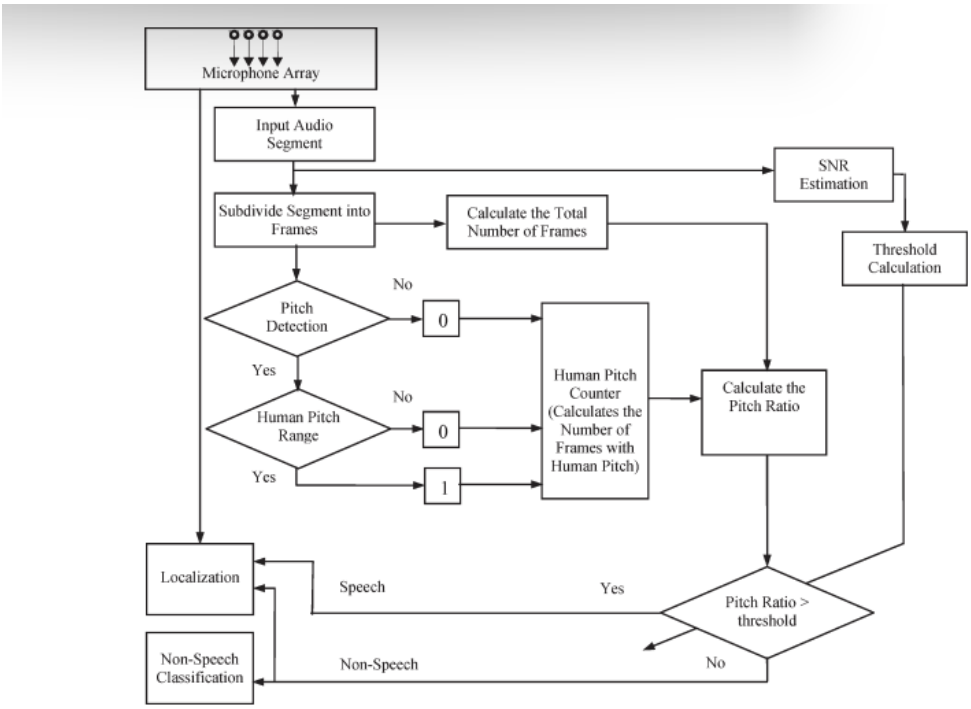
Uses an algorithm to classify an audio segment as speech or non speech. Does this by: **diving audio segments into frames, estimates pitch in each frame, and calculates the pitch ratio.**

Neural network is then used for further classification.

Approach:



1) ***For APR:*** Use Pitch Ratio to divide audio segments into frames and calculate PR parameters. If above(or below) a certain parameter, we can classify under certain noise.
    a) ***Proposed Algorithm:***



2) ***For NonSpeech(Useful):*** Use a time-delay neural network. Input feature consists of MFCCs, DMFCC, and PR values. MFCC and DeltaMFCC are vectors that are calculated, Pitch values are then added to the vector. This is what is applied to the TDNN.
    a) Had a database of 911 audio segments.

Paper:
https://drive.google.com/drive/folders/1o-3MSHW3M9j5vrl8aO1giJNYIahTdRtR?usp=sharing

==========

Research Paper: Mobile Microphone Array Speech Detection and Localization in Diverse Environments:
https://ieeexplore.ieee.org/abstract/document/9616168?casa_token=fWmOTe-GRaIAAAAA:4wBSPwGpfPKCuMr0EQmic9tNV7OaBjZcJ7AIjaXsgJiYWvVzVgLOcxXRszX8oZRrZva-cE1I

Abstract: SELD: Sound Event Localization and Detection.
Uses convolutional recurrent neural networks.
Used mobile phone.

Approach(Sound Classification):

"*Sound classification is done by feeding time-frequency acoustic features extracted from audio to a deep neural network that estimates the class probabilities. Feature extraction and classification are elaborated below"*

1) Feature extraction
2) Estimation w/ probabilities
3) Deep Neural Network: Utilizes CRNN;

==========

Research Paper: Environmental Sound Classification with Convolutional Neural Networks.

Piczak2015-ESC-ConvNet.pdf (karolpiczak.com)

Abstract: Information on neural networks and their use in Sound Classification

==========

I wanted to read this but I dont have access…
https://ieeexplore.ieee.org/abstract/document/7502768?casa_token=4rd3pG3-RBEAAAAA:ieJA_6_DmgBBgHRJJx9jvOSJQ42P_Xg4831CeAuAZz5U5MMjvQzK34bLaF2W9GsZ73Wt_Z6Ppg

==========

Practical Application

Initial Noise Detection on Arduino (Uses MATLAB):
KADUKUNTLA-THESIS-2019.pdf (txstate.edu)

Identification of Mosquitoes using Wing Sound Classification on a Tiny Embedded System(Uses TinyML & Arduino Nano Sense):
IEEE Xplore Full-Text PDF:

Abstract:

Collects audio data from mosquitoes and classifies it using TinyML.

Possible Approach:

1) Used 'Edge Impulse' which does the DataBase, NN Training, Testing, Deployment, etc. **Free for developers(I think)**

2) Hardware interfacing, spectrogram and feature extraction are done on Arduino. The ML Model is applied and the arduino outputs results.

I.E: Work on the edge for the ML/NN then move the Arduino for things such as extraction, hardware, and interface/hardware (and to call the edge?).

Application: TensorFlow;
https://www.analyticsvidhya.com/blog/2021/06/introduction-to-audio-classification/

https://www.tensorflow.org/lite/microcontrollers#supported_platforms

==========

Requirements/High-Level

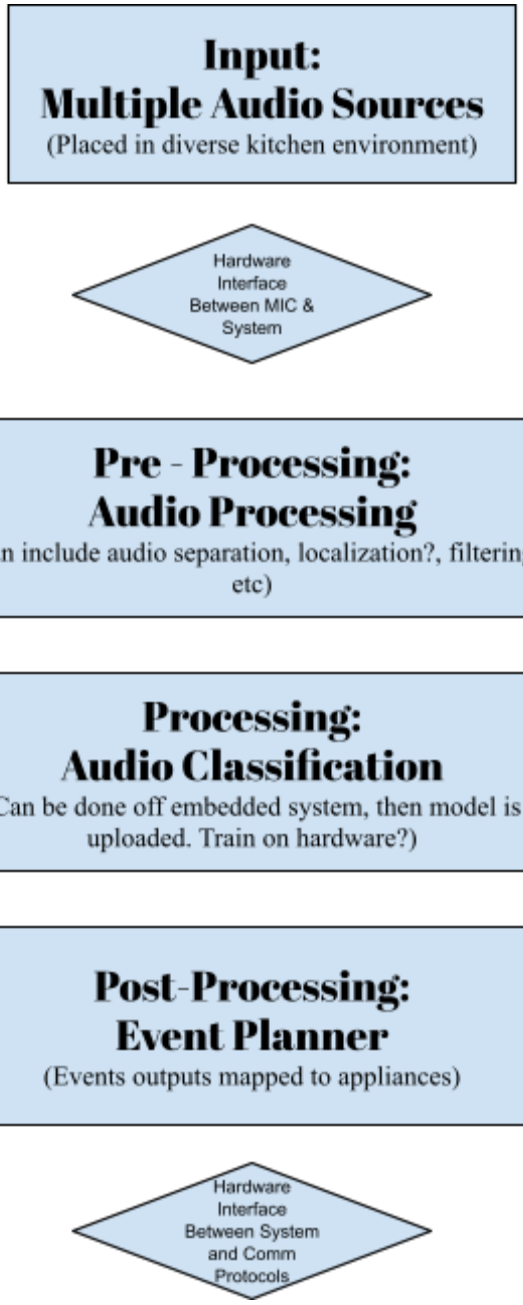
Goal:

Smart homes are essential conveniences for consumers who want to make their lives just a little bit easier. At the forefront of these advancing technologies are smart assistants. This project aims to advance smart assistant functionality in noisy environments, such as a kitchen, and be able to help further ease a consumer's life by identifying/classifying differing sounds while managing useful events for each noise identified.

Requirements/Objectives:

1) Identify if an appliance is on or off.
2) Classify the appliances being used.
3) Identify how long an appliance is on for.
4) Create an event or sequence of events based on appliances being used.
5) Identify multiple appliances being used at once (noisy environment).
    - STRETCH GOAL: Use a camera to help identify objects inside of the scene as well.
    - STRETCH GOAL: Send appliance 'stats' to the user. I.E: Potential Power Draw, Time On, Etc.

Top Level Design:



==========

==========

Hardware Platforms

Microcontroller Considerations and Requirements:

- Requirements:

        - Must be compatible with Edge Impulse and TinyML
        - Must have WiFi and Bluetooth support
        - We expect to keep the board plugged in (typical of smart assistants), letting us opt for low latency, high power boards
    - *Commonly used microcontrollers for AI/ML:* (https://www.seeedstudio.com/blog/2019/10/24/microcontrollers-for-machine-learning-and-ai/)
    - *Hardware Platforms for Neural Networks and Deep Learning(Research):* (https://arxiv.org/pdf/2108.09457.pdf, https://arxiv.org/pdf/2205.10369.pdf)

- **Board specific research and justification**
    - We opt for a CPU/GPU based board for better model latency.
    - **Jetson Nano**
        - https://developer.nvidia.com/embedded/jetson-nano-developer-kit
        - **Upside:**
            - CPU (runs anything you can run on your computer that fits within 4GB RAM)
            - Good performance
            - Great software/library support
                - Tensorflow, PyTorch, Caffe, Keras, MXNet support
                - NVIDIA TensorRT Accelerator Library
            - Comes with heatsink etc.
            - Has GPU for training models
                - FP GPU acceleration
        - **Downside**:
            - 4-year old SOC, decent all-round performance but not the most efficient
            - Large for embedded smart home/appliance
            - Separate Wifi dongle required
        - **Cost**: $99+
    - ~~**Coral Dev Board**~~
        - ~~https://coral.ai/products/dev-board/~~
        - ~~***Upside:***~~
            - ~~Newest chip, most efficient~~
            - ~~Best choice (high performance per watt)~~
            - ~~Wifi and Bluetooth support~~
            - ~~Low power microcontroller~~
            - ~~Built-in GPU for training models~~
        - ~~**Downside:**~~
            - ~~Locked into using only TensorFlow Lite~~
            - ~~Limited ecosystem~~
            - ~~Not fully supported by Edge Impulse~~
        - ~~**Cost**: $149+~~
    - **Raspberry Pi 4 B**
        - https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
        - **Upside:**
            - Most cost effective (cheap and decent performance)

        - Abundant documentation, great tutorials for most stuff
        - CPU (run any models that can be run on CPU)
        - Works with most ML frameworks
        - Can pair it later with a USB accelerator(TPU)
        - Great for CNN(If paired with Coral RPU)
    - **Downside:**
        - Depending on the type of model being ran, will not be powerful enough(RNN)
    - **Cost**: $35+ (increases per GB)
- **Raspberry Pi Zero 2**
    - ~~—~~Almost same as Raspi 4 (RAM and CPU speed limitations)
- ~~**STM32 Based Platform (not enough performance gains, we like python)**~~

Microphone Array Considerations and Requirements:

- Number of Microphones
    - More microphones, more accuracy
        - Requires more computational power and increases computation time
            - Shouldn't really be an issue with chosen boards, especially if GPU
        - https://sites.tufts.edu/eeseniordesignhandbook/files/2019/05/ECE_SHP_Final_Hoder_James_v01.pdf
        - *Dialnet: Analysis of Sound Localization and Microphone Arrays Paper*
            - 2 microphones is sufficient for low noise environments where the sound source is close to the array
            - 4 microphones increases ability of sound source detection in noisy environments with farther sound sources
    - Commercial microphone arrays typically have 2, 4, 6, 7 microphones in their configurations
- **Compatible Arrays with Boards**
    - **https://www.seeedstudio.com/ReSpeaker-Mic-Array-v2-0.html**
    - https://wiki.seeedstudio.com/ReSpeaker_6-Mic_Circular_Array_kit_for_Raspberry_Pi/
    - https://wiki.seeedstudio.com/ReSpeaker-USB-Mic-Array/
    - https://www.matrix.one/products/creator
    - https://andreaelectronics.com/nvidia-jetson-microphones/
    - Many have localization algorithms to start off with
- Flexible Placement
- Board Space
- Reliability
- Configuration/Geometry
    - Linear, Rectangular, Triangular, Circular, Spherical, Omnidirectional
        - Based on the geometry, DOA algorithm will also change
        - https://www.scitepress.org/Papers/2013/47857/47857.pdf
            - ULA vs URA vs UCA (linear vs rectangular vs circular)
            - UCA geometry allows for better use of localization in a 3d space
            - Accuracy of ULA-4 not sufficient in beamforming process

        - ULA-11 gave best accuracy
        - URA seems to reduce the angular resolution??
    - Ad Hoc
        - Multiple single microphones (arduinos?)
        - Increases complexity in sound source calculations
        - Not an easily generalizable system
- Matching (microphones in array must be similar and closely matched if not identical)
    - Directionality
    - Sensitivity
    - Phase
    - https://www.vissonic.com/news/what-to-look-for-in-microphone-array-matching.html#:~:text=Therefore%2C%20the%20three%20aspects%20of,are%20directionality%2C%20sensitivity%20and%20phase.
    - Only an issue with ad hoc microphone array configuration

NOTE: when comparing the unidirectional microphones, the terminology “cardioid microphone” was used due to the range of the sound that it picks up.

Omni-Directional Microphone

http://www.learningaboutelectronics.com/Articles/Types-of-Microphones

- Omnidirectional microphones pick up unwanted noise
- Unidirectional arrays are well suited to picking up from one direction (kitchen)

https://www.syncoaudio.com/blogs/news/omnidirectional-microphone-vs-unidirectional

- Leakage in omnidirectional but better performance
- Channel separation less precise in omnidirectional
- More extended frequency response in omnidirectional
- Proximity effect
    - “Proximity effect is **the exaggeration of low-frequency sounds in a directional microphone when the microphone is located very near the sound source**. Proximity effect has an influence on the frequency response of a directional microphone.”
    - Not really an issue for us since the microphone will be a considerable distance away from the appliances
- Omnidirectional are less sensitive to pop and wind noises
    - Not an issue again since we have flexible placement indoors
- Omnidirectional microphones tend to pick up ambient noise more easily and more noise in unwanted directions as well

https://www.presentationproducts.com/a-comparison-of-conference-room-microphone-solutions/

- Unidirectional microphones have a higher signal to noise ratio which allows us to put a threshold on ambient and background noise so that we can filter in wanted sounds whereas omnidirectional mics do not give us this flexibility

==========

Bhanu: NN/ML/Classification

Audio Classification Resources:

- For training purposes
  - https://github.com/mscuttari/embedded-sound-classifier/blob/master/docs/report.pdf
    - Board used: STM32F4 Discovery (with MP45DT02 MEMS microphone)
    - Recording audio sample through STM32 board and then their classification
    - Sequential feed forward neural network (works on 2 sounds)
    - Language: Python
    - Libraries: Tensorflow and Keras (X-Cube-AI)
  - https://docs.edgeimpulse.com/docs/tutorials/audio-classification
    - STM32 Discovery Board
  - https://aws.amazon.com/marketplace/pp/prodview-pnpmf6jpxxdag
    - SaaS
- **For Pre-trained**
  - **Dataset to use**
    https://research.google.com/audioset/dataset/index.html

- End to End execution
  https://blog.tensorflow.org/2021/09/TinyML-Audio-for-everyone.html

- **Audio Input - Appliances output**
  https://github.com/yinkalario/General-Purpose-Sound-Recognition-Demo
  -Python
  -Pre-trained data sets
  -CNN
  -Storage Space Needed: 280MB
  -RAM Needed: 300MB
  -700% CPU

- **Models:**
  https://zenodo.org/record/3987831

- **Approaches:**
  1. **Train the model on our own**
  2. **Use an existing model and modify it**
     a. **This is what we are preferring**

TinyML for Acoustic Classification
Pruning the Model

Take fixed sound stream and feed to NN.

Using the beam forming focus on one sound.
Separation of audio
Two identical sources?

==========

Localization

==========

- **To get tensorflow-lite running**

[How to Perform Object Detection with TensorFlow Lite on Raspberry Pi (digikey.com)]
[Raspberry Pi Tensorflow Lite: Image classification and Object detection - Easy guide (survivingwithandroid.com)]
[examples/lite/examples/audio_classification/raspberry_pi at master · tensorflow/examples (github.com)]

Goals 11/4/2022:

- Dig into localization code
  - https://wiki.seeedstudio.com/ReSpeaker-USB-Mic-Array/
  - https://respeaker.io/4_mic_array/
  - https://github.com/respeaker/mic_array
    - https://arxiv.org/pdf/2103.03954.pdf
    - http://ras.papercept.net/images/temp/IROS/files/1414.pdf
- Tensorflow lite image and audio classification combination of two;
- Extract data from audio classification
- Beam forming; Can we focus on a location before an instance of audio? If not, can we focus on a location when there is a small instance of noise?

==========

ODAS

==========

ODAS instructions:

The process to get a fully functional and useful ODAS is two-fold.

1. ODAS
   a. Install and build

```
sudo apt-get install libfftw3-dev libconfig-dev libasound2-dev
libgconf-2-4 libpulse-dev
sudo apt-get install cmake
cd ~/Desktop
git clone https://github.com/introlab/odas.git
```

```
mkdir odas/build
cd odas/build
cmake ..
make
```

There might be some missing libraries, just sudo apt upgrade and update, sudo apt install them, sudo apt upgrade and update again, and build again.

b. Configuration
   i. Feel free to look at this: https://github.com/introlab/odas/wiki/configuration
   ii. But it worked for me just by using the file in **PATH/odas/config/odaslive/respeaker_usb_4_mic_array.cfg**
      1. When you see something like this:
         interface: {
               type = "socket"; ip = "127.0.0.1"; port = 9004;
               #type = "file"; path = "postfiltered.raw";
            };
         KEEP THE SOCKET, should not be file (cuz we aren't using recordings)
      2. For dynamically focusing the microphone at changing noises: (From github issue, using dynamic sst with tags, filter at output https://github.com/introlab/odas/issues/13#issuecomment-816457955)
         a. For sound separation, change **`mode_pf = "ms"`** to **`mode_pf = "ss" (in sss)`**
            *****we can change gain parameters to try and change the focus of the source (im just guessing)*****
            i.

```
ss: {
        Gmin = 0.0001;
        Gmid = 0.9;
        Gslope = 20.0;

    };
```

         b. Using tags with dynamic sst:

```
target = (
{ tag = "myTrackedSource"; x = 0; y = 1; z = 0}
);
```

         OR maybeeee

```
target: (
{ tag = "z"; x = 0.0; y = 0.0; z = 1.0},
{ tag = "x"; x = 1.0; y = 0.0; z = 0.0}
```

```
        );
    OR maybeeee
        target = (
        { tag = "myTrackedSource"; x =
        x-coordinate; y = y-coordinate; z =
        z-coordinate }
        );
```

c. Running ODAS (on its own)

```
./odaslive -c myConfigFile.cfg
```

Where myConfigFile is respeaker_usb_4_mic_array.cfg**. Notice how it is in the same directory as the odaslive file. The odaslive file will be in **PATH/odas/build/bin/live**. If you do not want them in the same directory just make sure the paths match up when calling odaslive.

2. ODAS WEB (also called STUDIO or THE GUI)
   a. Install and get and node

```
su
```

You may need to downgrade python.

```
sudo apt remove python3 -y
sudo apt install python2 -y
```

   b. Configuration
      i. Make sure the config file has the following SSL info:

```
potential: {
  format = "json";
  interface: {
    type = "socket";
    ip = "<IP>";
    port = 9001;
  };
};
```

      ii. Make sure the config file has the following SST info:

```
tracked: {
  format = "json";
  interface: {
    type = "socket";
    ip = "<IP>";
    port = 9000;
  };
```

```
};
```

iii. Make sure the config file has the following SSS info:

```
separated: {
  fS = <SAMPLE RATE>;
  hopSize = 512;
  nBits = 16;

  interface: {
    type = "socket";
    ip = "<IP>";
    port = 10000;
  }
};

postfiltered: {
  fS = <SAMPLE RATE>;
  hopSize = 512;
  nBits = 16;

  interface: {
    type = "socket";
    ip = "<IP>";
    port = 10010;
  }
};
```

c. Running ODAS with ODAS WEB

```
cd PATH/odas_web
npm start
```

d. On The GUI
   i. Change the Local System Monitor to be the IP of the Raspberry Pi.
   ii. In the ODAS Control section:
      1. Select ODAS core (PATH/odas/build/bin/odaslive) under the ODAS core section
      2. Select config file (PATH/odas/config/odaslive/respeaker_usb_4_mic_array.cfg) under the ODAS config section
   iii. Hit Launch ODAS.

*****ONLY FOR RESPEAKER 4 MIC ARRAY*****
u need something called arecord, make sure it is installed, you can check with this:
arecord -l
It will output something like this:

****** List of CAPTURE Hardware Devices ******
card 3: seeed4micvoicec [seeed-4mic-voicecard], device 0: bcm2835-i2s-ac10x-codec0 ac10x-codec0-0 [bcm2835-i2s-ac10x-codec0 ac10x-codec0-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
Then, change the sound card designation in 18th line, in the config file (card 3 as shown above)
open odas web, select the control kernel an config file to read in 'odas control'
*****STOP IGNORING*****

==========
MongoDB
==========
https://www.mongodb.com/developer/products/mongodb/mongodb-on-raspberry-pi/
I will make a mock db on my computer first to get an example running
Then I will create a service on the cloud to ping the device and get db info

Python script that sets off following pipeline:
Starts mic array input + camera input, model and db conn → model detection → db write
Python script:

```
db =
connect("localhost:27020/myDatabase");
```

https://stackoverflow.com/questions/36942980/running-two-executable-in-parallel-with-os-system-in-python

```
import thread

thread.start_new_thread(os.system, ('a.exe',))
thread.start_new_thread(os.system, ('b.exe',))
```