Appliance Detection

Smart Home Automation Project
Embedded Systems
By: Disha Zambani, Bhanu Chaudhary, Nathan Portillo

Motivation

Background:

Smart homes are essential conveniences for consumers who want to make their lives just a little bit easier!

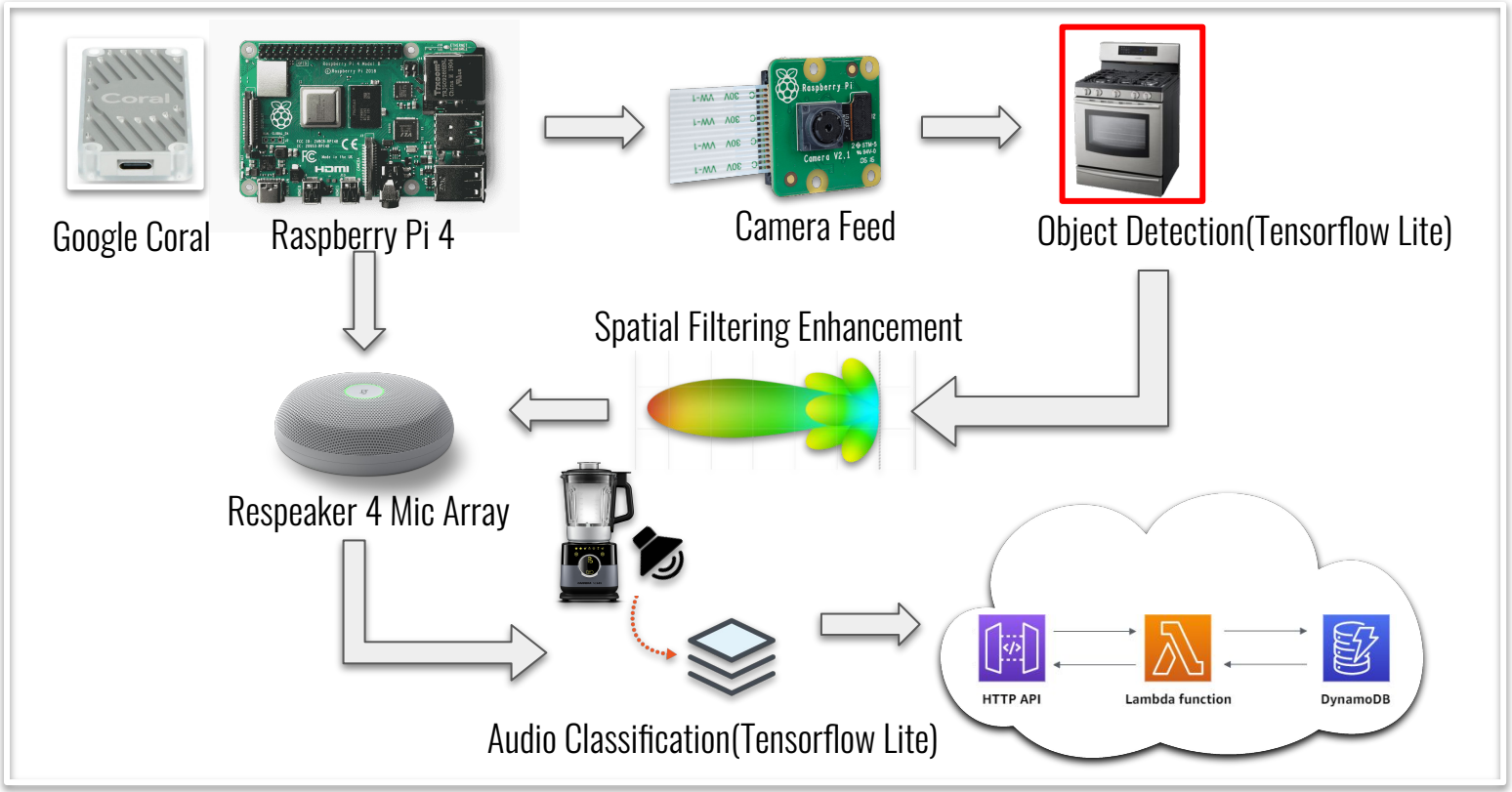
At the forefront of these advancing technologies are smart assistants:

amazon alexa

Goals:

- Advance smart assistant functionality in noisy environments.
- Ease a consumer's life by identifying/classifying differing sounds and appliances.
- Manage useful events for each noise identified.

High-Level Approach



ODAS/ODAS_WEB

✓ Features:

- Sound Source Separation
- Sound Source Localization
- Sound Source Tracking
- Post Filtering

Features of Interest:

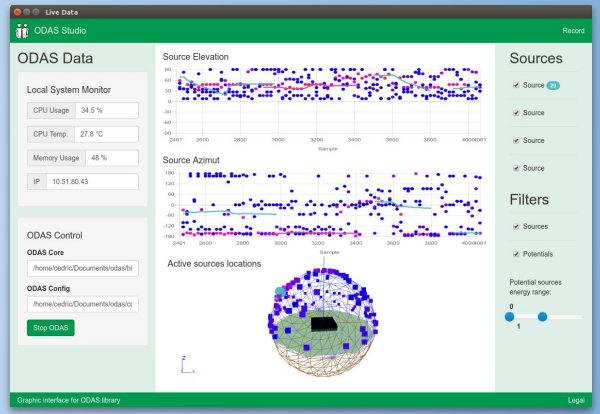
- Static Sound Source Tracking
- Spatial Filters

Initial Approach:

- Use the Static Sound Source tracking to beamform to locations of interest dynamically.

Secondary Approach:

- Use Spatial Filters



✗ Certain features broken(Static Sound Source tracking does not always work)

✗ Difficult to output/use data without ODAS_WEB; Faced many difficulties running ODAS_WEB

✗ Dynamically configuring file at runtime required restructuring of project or scripting

Object Detection

MODEL

- Tensorflow Lite Pre-Trained Model
  - Based on COCO Common Objects Dataset
  - Streamlined:

ENHANCEMENT FEATURES

1) Centroid Extraction:

```
{'oven': (444.0, 400.0), 'sink': (137.5, 429.5),
{'oven': (444.0, 402.0), 'sink': (137.5, 429.5),
{'oven': (444.0, 402.0), 'sink': (137.5, 432.5),
```

2) Closeness Detection:

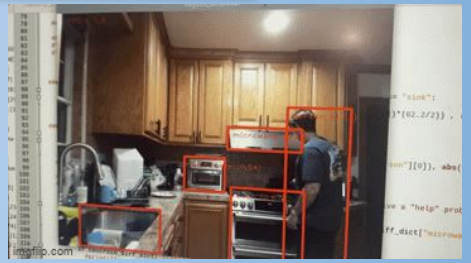
```
if "person" in centroid_dict.keys():
  for key in centroid_dict.keys():
    centroid_diff_dict[key] = (abs(centroid_dict[key][0] - centroid_dict["person"][0]),
    abs(centroid_dict[key][1] - centroid_dict["person"][1]))
    #print(centroid_diff_dict)
```

3) Angle Detection:

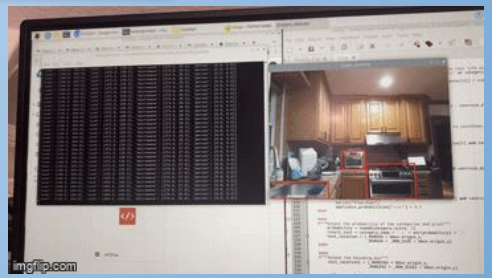
```
if category_name == "microwave" or category_name == "oven" or category_name == "sink":
  for x in centroid_dict.keys():
    angle_from_center[x] = ((((centroid_dict[x][0] - ccenter[0]) / ccenter[0])*(62.2/2)) ,
    ((centroid_dict[x][1] - ccenter[1]) / ccenter[1]) * (48.8/2))
    print(angle_from_center)
```

OUTPUTS

Object Detection/Centroids:



Closeness Detection:



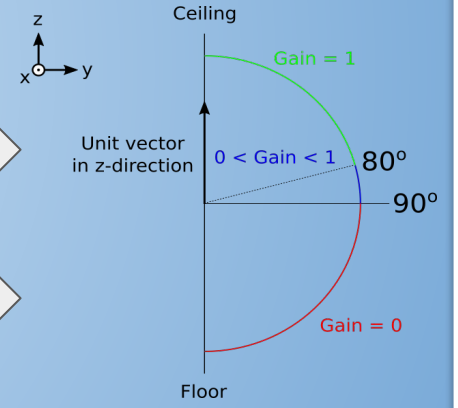
Microphone Array

STANDARD IMPLEMENTATION

- Listen for Noise Instance
- Dynamic Beamforming
- Isolate & Focus
- Localization, DOA, etc.

PROPOSED IMPLEMENTATION

- Listen for Noise Instance
- Angle & Closeness Detection
- Dynamic Spatial Filter
- Dynamic Beamforming



Microphone Array

STANDARD IMPLEMENTATION

PROPOSED IMPLEMENTATION

```
spatialfilters = (

    {
        direction = ( +0.000, +0.000, +1.000 );
        angle = (80.0, 100.0);

    }
    {
        direction = ( +1.000, +0.000, +0.000 );
        angle = (angle_detection_x + 10.0 , angle_detection_x +20.0);

    }
    {
        direction = ( +0.000, +1.000, +0.000 );
        angle = (angle_detection_y + 10.0 , angle_detection_y +20.0);

    }
```

Audio Classification

MODEL	ENHANCEMENT FEATURE	OUTPUT
Tensorflow Lite Pre-Trained Model - Based on Google AudioSet Data - Extracting Required Objects	Confidence Level Normalization Hysteresis	Identification of the Devices Event Based Decisions

Event Based Actions

FOOD CHOPPING	Parent Process starts a child process - music and kills it when chopping stops
BLENDER	Sends a message to the user if blender remains on for more than a specified time - using Twilio for this
WATER FAUCET	Sends a message to the user if water faucet remains on for more than a specified time - using Twilio for this
FIRE ALARM	If there is a fire alarm, we start a live feed from the camera that the user can watch in addition to sending message
FIRE	If we detect fire, we simulate a fire alarm - based on object detection

Music in Action

Message Notification in Action

User Interface

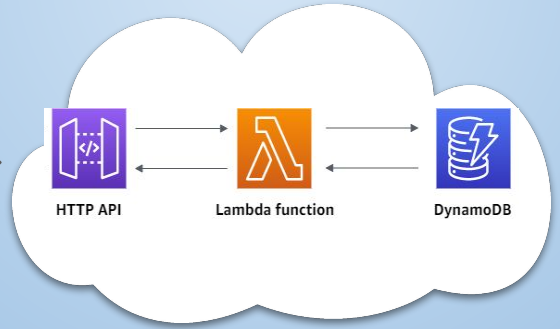
EVENT

When an appliance is detected:
{
appliance: “sink”,
status: True,
timestamp: datetime.datetime.now()
}

When an appliance stops running:
{
appliance: “sink”,
status: False,
timestamp: datetime.datetime.now()
}

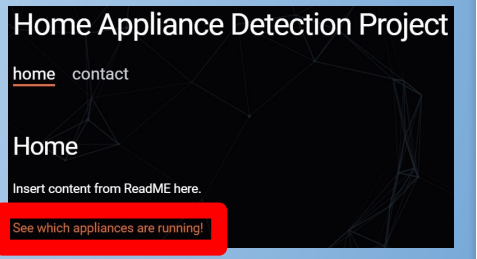
HTTP PUT

AWS



UI

HTTP GET



{"Items":[{"stat":true,"id":"Chopping (food)","timestamp":"12/08/2022, 01:08:19"},
{"stat":true,"id":"Water tap, faucet","timestamp":"12/08/2022, 01:34:50"},
{"stat":false,"id":"Blender","timestamp":"12/08/2022, 11:18:41"}],"Count":3,"ScannedCount":3}

Challenges

- Lack of flexible pre-existing localization techniques
  - Integration of ODAS with Python/Re-Speaker Configuration File w/ Dynamic Variables.
- Integration of Individual Components
  - Finding features to take advantage of and implement.
  - Sequential vs Parallel Processing
- Chip Shortage - Lack of PI available

Future Steps

- Implement parallel execution of models.
- Implement Stereo Camera! Can easily take into account depth/XYZ real-world coordinates.
- Create dynamically configurable localization:
  - Write a script that accesses configuration file and modifies as variables change.
  - Create spatial filtering from scratch.
- Add multiple users with credentials to view personal appliances
- Create specialized Audio and Computer Visions models from scratch
- Implement Live Feed Feature
- Develop a more intuitive and attractive UI

Contributions

- Bhanu
  - Audio Classification Model research
  - Model selection, tuning and normalization
  - Creating a separate process for parallel task
  - Twilio text updates
- Nathan
  - Model research
  - Localization and beamforming
  - Computer vision detection
- Disha
  - Hardware research
  - Localization and beamforming
  - Computer vision detection
  - AWS integration

References

- https://github.com/yinkalario/General-Purpose-Sound-Recognition-Demo
- https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/raspberry_pi
- https://github.com/tensorflow/examples/tree/master/lite/examples/audio_classification/raspberry_pi
- introlab/odas: ODAS: Open embeddeD Audition System (github.com)
- Spatial Filter configuration · Issue #19 · introlab/odas (github.com)
- https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html
- https://wiki.seeedstudio.com/ReSpeaker-USB-Mic-Array/
- https://www.w3schools.com/python/module_requests.asp

Audio Classification Research

- Different Approaches tried
  - Training our own models - Edge Impulse
  - Fine Tuning Pre-trained models - Using AWS SaaS
  - Fully Trained Models
    - General-Purpose-Sound-Recognition-Demo
    - Tensor_flow_lite_audio_classification