CSE237A - FINAL PROJECT REPORT

1. **TITLE**: **AUDIO BASED ACTIVITY RECOGNITION**

2. **MEMBERS**: Sivasankar Palaniappan -spalania@ucsd.edu, A53275703

Chandhini Grandhi – cgrandhi@ucsd.edu, A53272378

3. **MOTIVATION:** Smart home is a large and upcoming field, with vast number of applications. They are mainly divided into activity recognition and actuation/automation. Large amount of work has been done in the field of automation. Activity recognition on the other hand, is an upcoming field. Most of the work present today has been performed using smart watch or wearable sensor to detect basic activities. This can yield good results but involves large number of sensors, along with large amount of processing of data originating with different sensors present. Moreover, the cost is high and also power dissipated will be very large. The subject may be uncomfortable wearing too many sensors or wearables. A Fitbit is not affordable by all. Even with large number of sensors, some of the activities cannot be detected. A person may just be sitting but it is not possible to find out if he or she is drinking water or eating. To detect such activities, we propose a model of using microphone which will detect the noise or the sound and find out the activity using an appropriate training model. On the system end, using a raspberry pi makes the entire setup as portable, reducing space complexity and avoiding reducing cost as the system is cheaper than using a standalone CPU or a laptop.

4. **RELATED WORK:** In general, there are large number of academic works related to activity recognition using various sensors such as accelerometer, gyro meter, smart watch, Fitbit. To compare our work, with the previous work, only academic papers pertaining activity recognition using audio was considered.

   a. In this field, the amount of work done has been minimal compared to the method mentioned above.

   (i)- “Audio-Based Activities of Daily Living (ADL) Recognition with Large-Scale Acoustic Embeddings from Online Videos “performs audio detection but uses audio from online videos, and does not manually obtain data sets.

   (ii)- “A Similarity Analysis of Audio Signal to Develop a Human Activity Recognition Using Similarity Networks” uses mobile phones to obtain the audio files and obtains the activity using similarity analysis by calculating Euclidean Distances.

   (iii)-"Audio-based human activity recognition using Non-Markovian Ensemble Voting," manages to classify large number of activities by using a sound book which consists of bag of sounds (similar to bag of words) and detects activities using a voting process by comparing with entries from sound book.

   b. Our work is different from the current literature in the following aspects

   (i) Our project manages to obtain data using an inexpensive setup by just recording it ourselves without having to search for large number of video embeddings for audio of several activities.

   (ii) Performing a similarity network would take a large amount of time, since the Euclidean Distances have to be calculated which can be a highly time taking process. Instead we have performed a comparison of various classification-based machine learning algorithms, by considering accuracy and time to execute as parameters, which have not been done by the previous literature.

   (iii) Activity recognition based on audio has been done on general purpose systems and has hardly been done on a standalone microcomputer such as a raspberry pi which gives an advantage of being portable and performs all activities which can be done on a general-purpose CPU.

5. HARDWARE COMPONENTS:

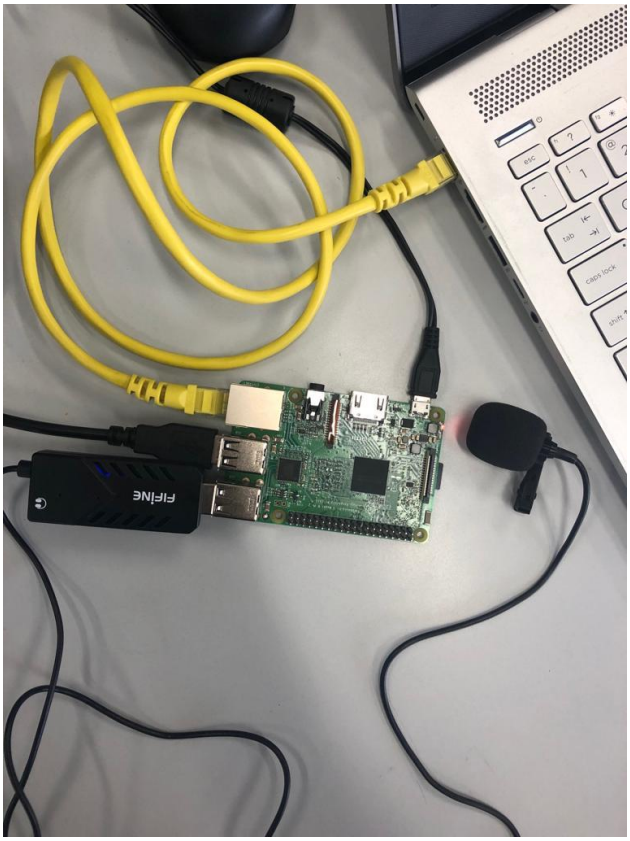
Raspberry Pi and microphone – Less and cheap hardware is the novelty of our project. We have been able to successfully connect the microphone to the raspberry pi and take clean audio samples.

Microphone: We used Plug and Play USB Microphone which can be bought from here. Raspberry Pi:

Raspberry Pi3 Model B Quad-Core 1.2 GHz 1GB RAM which can be bought here

Power Supply for Raspberry Pi: Official Raspberry Pi Power Supply of 5V, 2.5A which can be bought here.

Below is the picture of the Hardware setup we used:



Hardware Design choices:

- We used this because it was the least expensive and supports a wide range of frequencies (50-16000 Hz) for sound reproduction.
- We chose Raspberry Pi Model 3 as we wanted to create a portable system which can be used anywhere with all features of a normal CPU. Moreover, large amount of literature is present for using Rpi for IOT and hence we felt that raspberry pi would be a suitable choice for our project. As data keeps increasing, overhead has become a major issue and hyperdimensional computing (HD) is a growing field to counter this problem. Audio is a large data-based parameter and training this on a single system can be a hindrance. So, a RPi can be used as a part of HD for future purposes.

- We chose the official power supply to prevent under volting since we are dealing with large amounts of data, to be on the safer side.

6. SOFTWARE COMPONENTS:

- Python – Version 3.5
- Raspberry Pi3 Kernel – Raspbian (Debian) version 4.14
- Scikit-Learn – Version 0.21
- Numpy – Version 1.10.1
- TKinter – Version 8.6
- Alsa Audio –Version 1.1.7
- Jupyter Notebook-Version 5.7.4
- IDLE- For Python 3.5

Software Design Choices:

- Since we were implementing machine learning algorithms we decided to use a programming language which had libraries for doing the same. Since we were working on audio we needed signal processing libraries as well. MATLAB and Python were two programming languages which implemented both but for a raspberry Pi it was evident that Python was a better choice as the Debian had a pre-loaded python software which was free and took lesser space of the memory while MATLAB was paid software which was very large in size (2GB of HDD, needing 4-6GB for installation). Hence, we decided to go with Python.
- We used the latest kernel in order to use the latest and most available features and also that we would have better technical support, in case we ran into any issues.
- We used the default python libraries for machine learning (Scikit learn), numerical computations (Numpy), GUI (tkinter) ,due to their ease of use and simplicity and large number of features and documentation which were available.
- Alsa Audio is Linux's default audio library which lets to interface devices, record and play sound using command line.
- Jupyter Notebooks were used due to the easy and comfortable UI, which makes subdivide the code into several cells and helps in the debugging process as well.
- IDLE was used for running the completed code as it seemed more professional for running the GUI.

7. INTEGRATING HW AND SW:

The novelty of our project was providing a system with minimal hardware and we interfaced a microphone with the raspberry pi. The microphone was connected to the USB port of the Raspberry Pi and the microphone was configured by finding out the device number, name and card number by using the arecord and aplay commands which are part of the ALSA library. Using these the configuration file was written to set the microphone as the default playing and recording equipment. (Present as .asoundrc file in the code). Despite having a single hardware we faced difficulties as follows:

-We found the recording from the microphone to be very noisy or static after setting it up. We found that the issue was that we were speaking very close to the microphone.

- Using the microphone we were able to record audio with the alsa library which took the default sampling rate of 48KHz and gave mono output. But for python we were not able to use the sound-device library due to the change in the sampling rate. So finding out the appropriate library or command to record or play an audio was a challenge.

- We could not use the pyAudio library as the latency of obtaining the audio was higher (due to continuously streaming using callback mode) than when using the alsa CLI command with subprocess library.

Since we had to collect audio samples from laundry, vacuum cleaner we had to use an ethernet to connect raspberry pi to move to various places and perform various activities. Although the ethernet was not technically a hardware device part of our project, we still faced significant difficulties and large amount of time in connecting the ethernet to the laptop and the raspberry pi and making them work. We solved the issue by reloading the OS (Raspbian). Also we gave a default IPV4 address in the file cmdline.txt in the boot of the OS to communicate with the raspberry pi.

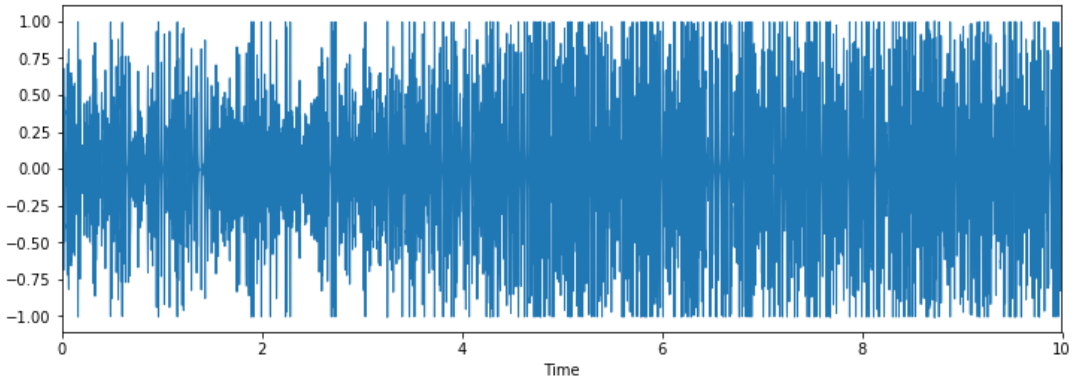
8. EXPERIMENTAL SECTION:

Data collection and Visualization

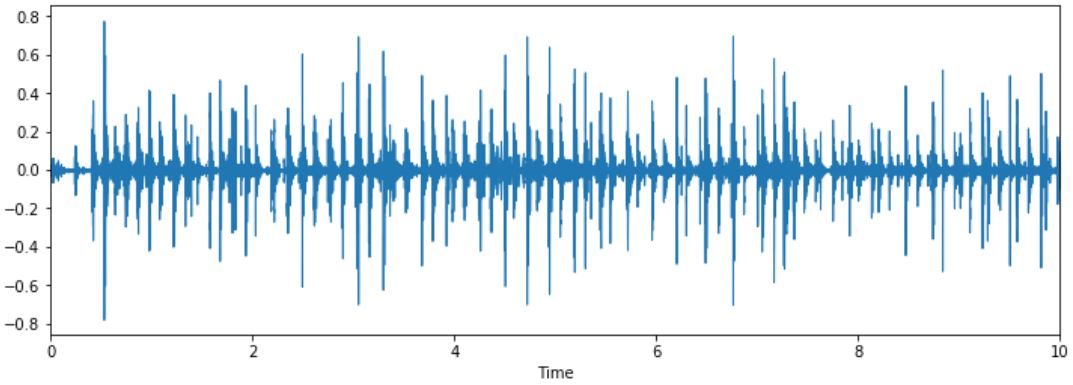
- We collected data for all 5 activities – Laundry, Eating, Hair drying, Vacuuming, Typing using the USB Microphone interfaced with the Raspberry pi
- We collected 50 audio samples of each activity for a duration of 10s. Below table represents the various type of data collected for each activity.

ACTIVITY	DATA COLLECTED
Hair Drying	Data was collected on all 3 speed and heat settings
Laundry	Data was collected on washer and Dryer, when they were running on varying speeds. (i.e – when the machine switched on vs when the machine is running at full speed)
Eating	Data was collected on all kinds of food- soft and hard. Particularly we collected data when the subject was eating Chips, Bread, Apple, Grapes, Chocolate, Cookie, Chewing Gum.
Vacuuming	Data was collected on all setting on the vacuum cleaner- low to high.
Typing	Data was collected when the subject was typing rigorously, when the subject was typing slowly and when the subject is typing and using mouse simultaneously.

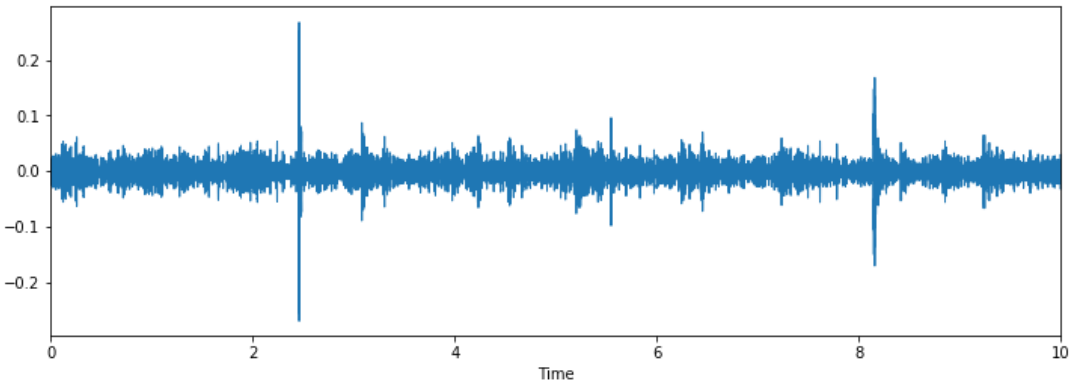
- We then visualized data by plotting them using Python to see if they were actually different and if we can run a classifier to classify them accurately. Below are the plots for each activity.
    - Hair Drying



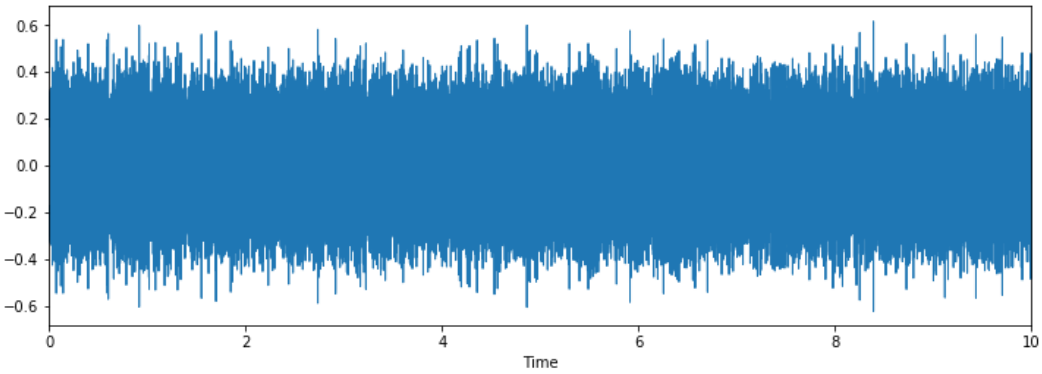
- Typing



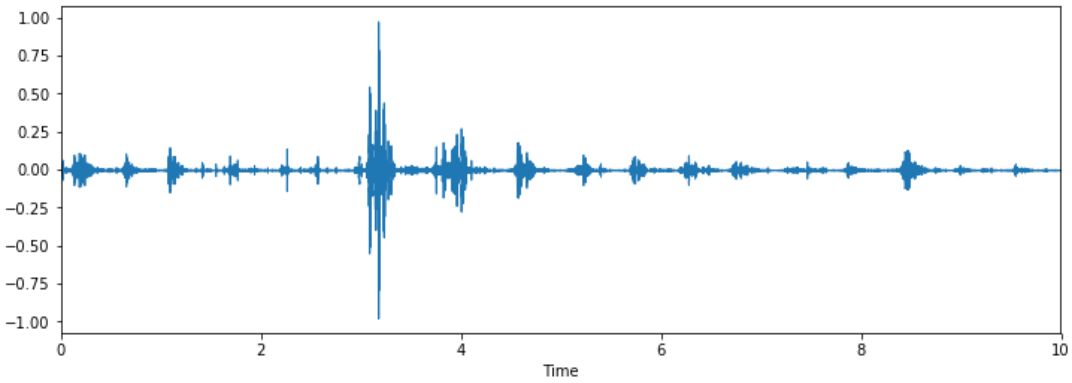
- Laundry



- Vacuuming



- Eating



It can be seen from the above plots that the audio signals have different principle components and can be classified with a classifier.

Feature extraction and building Feature Vector:

For the feature, since audio was being used we decided to use a feature in the frequency domain. Based on literature we found that for both audio and speech recognition MFCC or mel-frequency cepstral coefficients were extensively used. This works on the concept of human hearing , which can differentiate better in the mel frequency scale rather than the normal frequency scale. Each sound has a particular shape and this shape can be accurately described as an envelope of a short time power spectrum which is given by the MFCCs. They are generally used for Automatic Speech recognition , but however they can be used for other sounds as well.

The MFCCs are calculated as follows:

1. Divide the audio into several frames (0.25 seconds with overlap of 0.1s)

2.Calculate the periodogram estimate of the power spectrum

3. Apply the mel filter banks by taking product of both the energy and the mel banks

4. Taking logarithm of the entire value and computing Discrete Cosine Transform (DCT)

5. Taking the first 13 features, and discarding the rest

Apart from the MFCC to improve on the accuracy delta and d-delta components are used. Delta is the differential of the first frame of MFCC with the second frame and so on. D-delta is the difference of the delta components.

The audio is divided into several frames as the audio tends to kind of stationary in small time scale and hence this property can be exploited by dividing into frames. The power spectra is calculated as the human ear (cochlea) vibrates differently at different powers based on the frequency and this can be used to differentiate the sound. However they cannot differentiate between minute differences in frequencies or sounds for which the the mel banks are used. The power periodogram is put in different bins and summed together. At higher frequencies the variations of energy are not important and hence the mel scale is used which provides in proper spacing of the bins.

The formula for converting from frequency to Mel scale is:

$$M(f) = 1125 \ln(1 + f/700) \qquad (1)$$

To go from Mels back to frequency:

$$M^{-1}(m) = 700(\exp(m/1125) - 1) \qquad (2)$$

The mel banks are computed as follows:

1. Choose lower frequency as 0Hz and upper frequency as 24Khz (fs/2)

2. Convert them to the mel frequency scale

3. Choose 20 evenly spaced points between them for the 20 filter bank values

4. Convert them back to the frequency scale (h(i))

5. Find the corresponding FFT bins for the frequency resolution as f(i)= (fftno*h(i)) /samplerate- here the fft number is the fft taken during the calculation of the power which is calculated using the sample rate and window length. Here fftno=2048.

6. Obtain the mel banks using the following equations,

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

Find the product of the mel banks and the power values and take the logarithm as the human ear hears better in the log scale compared to the linear scale. DCT is taken as the audio frames are overlapping and the discrete cosine transforms helps in exploiting the property of correlation. Majority of the data or the information is stored in the first 13 values of the MFCC coefficients and hence the rest are discarded.

Delta and d-delta components give the variation of the MFCc and the delta coefficients respectively. They are obtained the padding the MFCC with 2 zeros at the beginning and at the end and finding the dot product of 5 components and finding the sum across all . This is given by the formula below

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

Here c=MFCC, N=2, (t+n to t-n corresponds to 5 values being taken for computation). Applying this formula for the delta coefficient, gives the d-delta coefficients.

In this project, we took 10s audio which gave us 1000 frames with each having 13 MFCC,delta and d-delta coefficients. They were all combined together to give a feature vector of size 39000. To train the models , 50 data samples of each activity were taken.

Machine Learning Algorithms:

Non- Real Time-based prediction:

- Now that we have the feature vector, we split the data in ratio of 75:25 for train and test respectively.
- We first implemented the Non-Real Time version of the prediction – i.e. We recorded the entire audio first, extracted the feature vector and ran the ML algorithm to predict the activity. The reason this is not real time is because it does not predict as the activity comes in.
- Since we are running the algorithms on Raspberry Pi, we wanted to use the classification algorithm that gives us the maximum accuracy and with least execution time.
- In order to find this out we ran all the classification algorithms and noted their execution time and accuracy and took an average of 5 runs for different combinations of the three key features – MFCC, Delta, DDelta.
- The below table shows the results:

Feature: MFCC

ML Algorithm	Accuracy	Execution Time (in seconds)
Logistic Regression	73.01%	95.17
SVM – Linear Kernel	66.67%	7.58
SVM -RBF Kernel	66.67%	110
K- Nearest Neighbors	79.36%	3.192
Decision Tree	100%	7.6
Multilayer Perceptron Classification	47%	23.28
Random Forest	100%	0.6
Boosting- Ada Boost	99.92%	185

Feature: MFCC and Delta

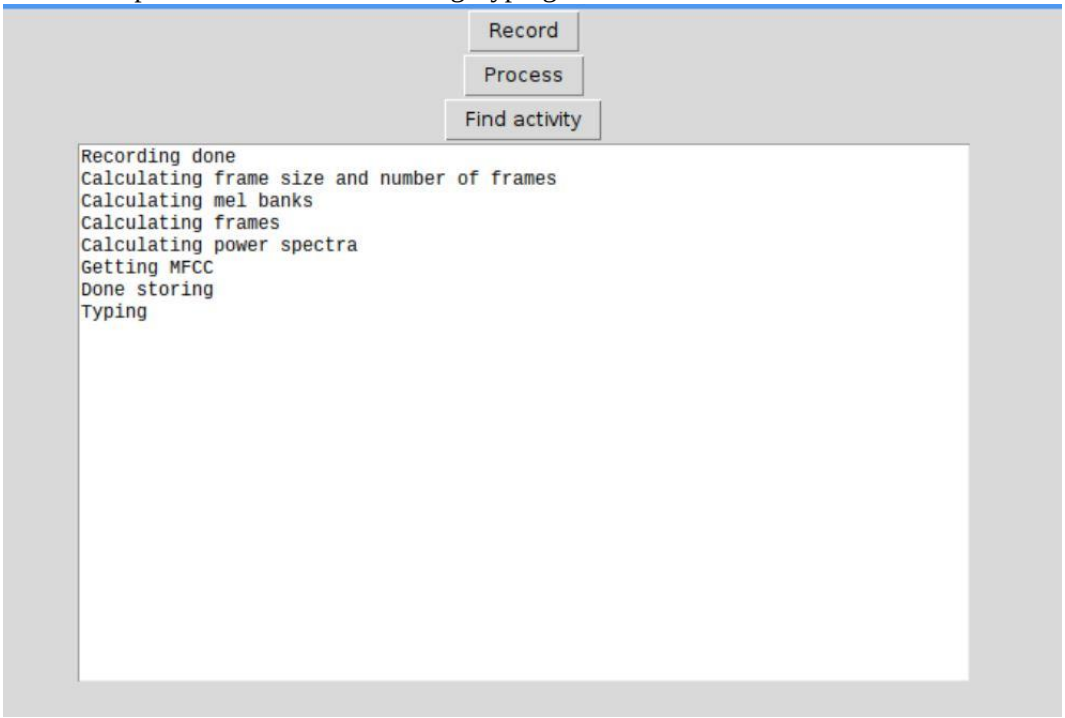
ML Algorithm	Accuracy	Execution Time (in seconds)
Logistic Regression	87.3%	218
SVM – Linear Kernel	77.77%	235
SVM -RBF Kernel	84.12%	14.7
K- Nearest Neighbors	77.77%	6.4
Decision Tree	100%	13.7
Multilayer Perceptron Classification	42%	46.53
Random Forest	100%	0.95
Boosting- Ada Boost	99.92%	365.9

Feature: MFCC, Delta and DDelta

ML Algorithm	Accuracy	Execution Time (in seconds)
Logistic Regression	88.85%	334
SVM – Linear Kernel	77.77%	439.213
SVM -RBF Kernel	90.47%	21.48
K- Nearest Neighbors	80.95%	9.82
Decision Tree	100%	20.91
Multilayer Perceptron Classification	53.96%	77.68
Random Forest	100%	1.25
Boosting- Ada Boost	99.98%	543.03

From the table, Random forest with one feature gives the best accuracy and execution time. We implemented the Non-real time-based activity prediction with one feature -MFCC and with Random Forest as our classifier algorithm.

- All the above steps were done in Jupyter Notebook and iPython Notebooks. To make the project more interactive, we developed a GUI using Python's Tkinter library.
- The GUI has three buttons for the following purposes:
    - Record – signals the Microphone interface to start recording for a duration of 10sec
    - Process – Opens the recorded audio file, builds MFCC feature vector by following the steps in Section 7 and stores the feature vector in a text file.
    - Find activity – Opens the text file, runs the Random forest algorithm to predict the label. Based on the label, it prints out the type of activity that was recorded.
- Below is a picture of the GUI detecting Typing

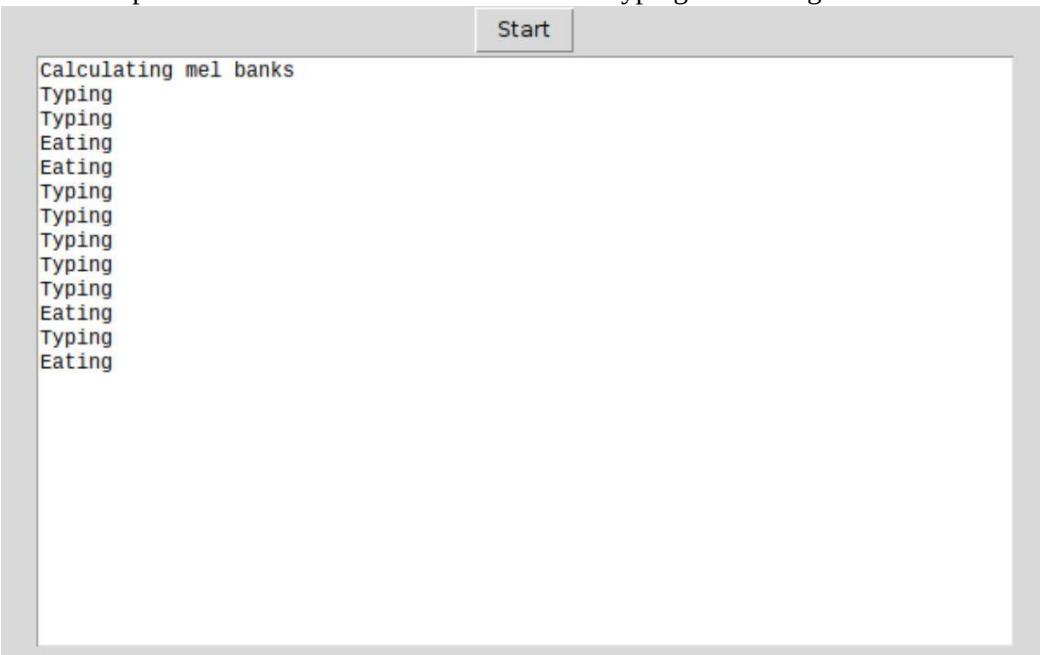


- During demo, we were not able to show the working of activities such as Laundry and Vacuum. Hence, we recorded a video and below are the links:
    - Laundry- https://youtu.be/0m3kBEFjWR4

- Vacuum- https://youtu.be/f9YrJ3E30Mk

Real Time-based Prediction:

- We implemented the real-time based prediction using multithreading by running two threads in parallel, one thread to record audio and one thread to run the Machine algorithm to predict the activity.
- We have a common queue which is shared between the two threads. The data that is being recorded is put in the queue by the Thread_One. Thread_Two consumes the data in the queue builds the MFCC feature vector.
- The built feature vector is then used to predict the activity by running Random Forest algorithm.
- Our main issue with this approach was that the algorithm was not predicting correctly and latency. The audio is 10s long and it is simultaneously calculating the feature vector as the audio comes in, because of this there was latency involved.
- To increase the probability of predicting correctly, we considered three features (MFCC, Delta, DDelta) for the real-time implementation as it will have more principle components for each label. This improved the prediction and our model was able to correctly classify the activities, but the latency issue persisted.
- All the above steps were done using Threading library in Python using Jupyter's Ipython notebook.
- The GUI has only one button:
  - Start – Starts the process by executing Thread_One and Thread_Two
- Below is a picture of GUI for a series of activities- Typing and Eating



- During demo, we were not able to show the working of activities such as Laundry and Vacuum. Hence, we recorded a video and below are the links:
  - Laundry- https://youtu.be/Ud1WwnJU2n0
  - Vacuum- https://youtu.be/93BD09TxFtk

Issues Faced:

- The main issue we had was in reducing the latency in real-time implementation.
- We implemented an approach of having a sliding window, which samples signals in blocks- i.e for a 9s audio signal, it samples first 3 seconds and builds the feature vector, then samples the next 3 seconds but it retains 1 second of the last frame and gets 2 seconds of the new frame in order to retain continuity.
- The above approach did not increase or decrease the latency, but it greatly affected the prediction as none of the activities were predicted correctly.
- We repeated the experiment 20 times and found the accuracy of having 10s audio frame in the real time to be as shown below:

Activity	Accuracy (in percentage)
Eating	80%
Vacuuming	75%
HairDrying	70%
Typing	85%
Laundry	65%

- We repeated the experiment 20 times and found the accuracy of the window based approach with samples of 3s

Activity	Accuracy (in percentage)
Eating	65%
Vacuuming	30%
Hair Drying	25%
Typing	55%
Laundry	35%

9. CONCLUSION:

a. We managed to accomplish the following

1. Create an inexpensive activity recognition system based on audio under 100$ by using raspberry pi of 45$ and microphone of 15$.

2. The system we created is portable and easy to carry from one place to another.

3. Despite using an inexpensive system we managed to create a system with an accuracy of greater than 75%.

4. Provide two different flavors of activity detection i.e by recording a sound and predicting it (non-real time) and also by predicting the activity as and when the sound is played. (real time).

5. Created two separate GUIs(one for real time and one for non-real time) for easy usage ie abstraction in the user end.

6. Compared various classification models of machine learning algorithms for the raspberry pi in terms of execution time and accuracy.

7. Perform the activity recognition using the best machine learning algorithm for raspberry pi based on results obtained in the comparison. Random forest gave maximum accuracy with the least time as seen in the results sections.

Real time implementation required the concept of threading which is an important concept of embedded systems where we had to record audio in one thread and process in the other thread. Also audio requires processing where we had to take the FFT to convert into frequency domain and then obtain the power spectrum and compute the DCT for getting the MFCC values. These are very important components of signal processing which is an important concept in embedded systems as there are large number of digital signal processors (DSP). DSP and threading are concepts which are discussed in the embedded systems class as well. Interfacing the microphone using USB is another important aspect of embedded systems as USB is a very important communication interface.

Overall the project as a whole is closely related to IOT (internet of things) which is nothing but a mixture of embedded systems, networking, data analysis, cloud and distributing computing. Sound is prevalent everywhere and this project uses audio as a commodity to predict the activity performed by the user. This project helps in the future development of embedded systems by contributing to the field of internet of things . Apart from activity recognition aspect of internet of things, this project also contributes to the field of hyper dimensional computing which helps in reducing the overhead of large data computation due to IOT and machine learning by splitting the computations to various dimensions. We provide help for this by computing the efficiency and accuracy of various machine learning algorithms which can help in providing a frame work for hyper dimensional computing.

b. The next steps will be the following :

1. Reduce the size of the dimensions of the feature vector by using principal component analysis to quicken the prediction

2. Try time domain audio features such as zero crossing rate, gradient, mean, centroid, standard deviation, periodicity of the energy spectra which will avoid computation of FFT and help in saving time. Apart from this, different techniques of FFT such as STFT can be tried for solving the above issue.

3. Try the window based approach in the real time prediction with varying audio sizes of 5s or 7s and find out which gives better accuracy and reduces latency.