CS-308-2016 Final Report

"Do Not Disturb"

10

Pratham Desai, 12D170001

Amal Dani, 120020019

Vishal Agarwal, 120110009

Darsh Shah, 120010010

Table of Contents

1. Introduction

Do Not Disturb is a tool that analyses the sound waves around and uses it as a feedback to alter the behavior of a music system in a desired way. The product is motivated by the lack of comfort caused by the music during conversations at various places. It is equipped to reduced the volume of the music system when the speech component of the environment increases. This facilitates our conversations and does not act as a mood spoiler. Another interesting use of the product is encountered when the same technology is used to assess the music played by the radio. It changes the channel when the radio jockey starts speaking. This prevents the driver, the pain of changing channels.

2. Problem Statement

Do Not Disturb is a tool that was developed to address the inconveniences caused during conversing in musical environments. Another use in our mind was to address the issue of drivers having to listen to the radio jockey. *Do Not Disturb* is a solution to these issues and we plan to achieve a decent accuracy in solving each of the issues. It solves the problem of inconvenient human intervention and tuning required to adjust the sound levels to optimal. It will come as a relief to the driver who is periodically bugged to change the music levels to match the current speech intensity levels.

We targeted 2 modes of operation:

- First mode: To enable changing of frequency automatically when the current radio channel is not playing music and the RJ is speaking or there is an advertisement playing
- Second Mode: To enable human voice dependent music volume change. The sound of music playing should reduce when there is some human voice in the surrounding and should go back to the original level when the volume reduces

3. Requirements

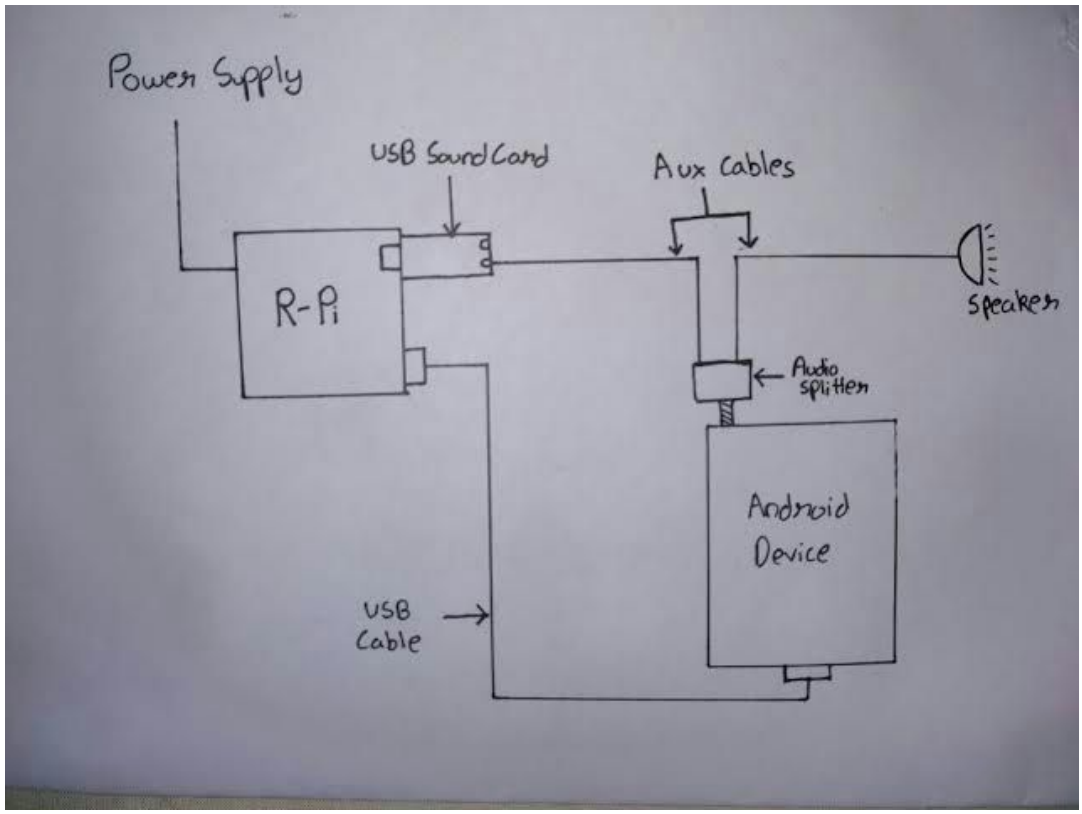
3.1 Harwdare Requirements

1) Raspberry Pi Model B+
2) Sound card
3) Android phone (only for prototype)
4) AUX cable
5) LAN cord
6) Speakers
7) USB cord

3.2 Software Requirements

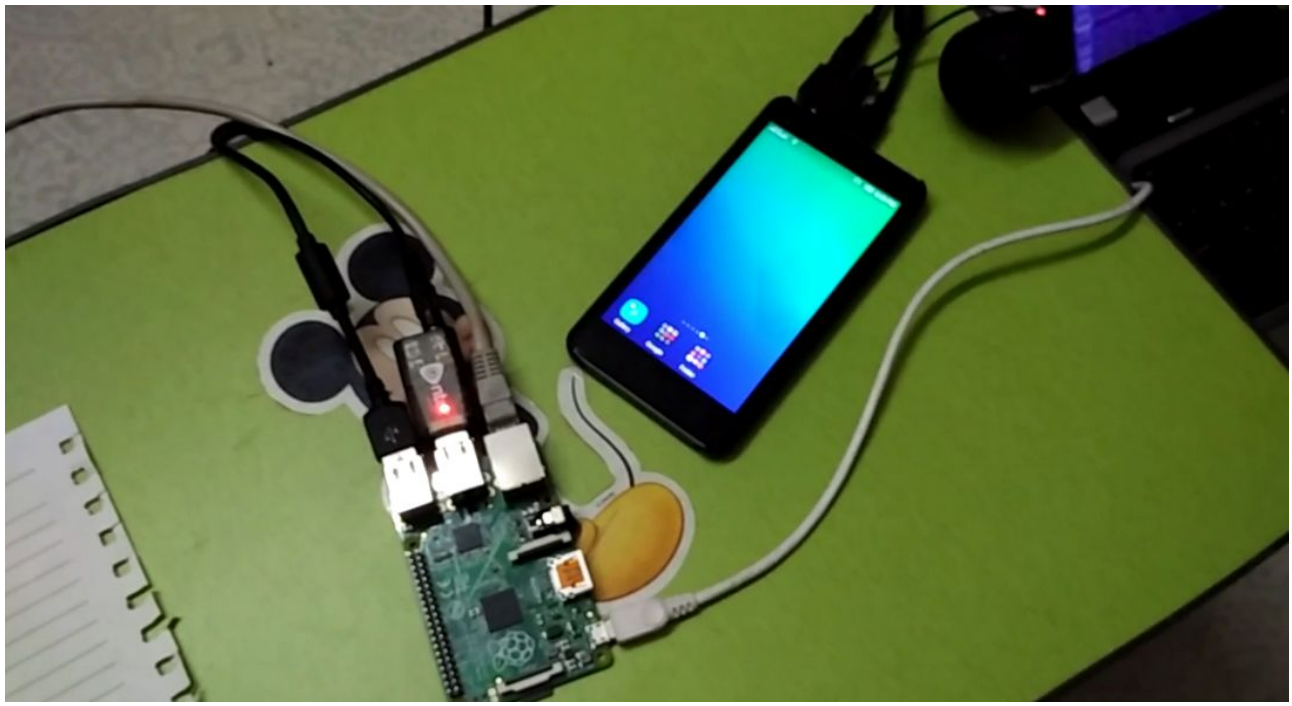
1) pyAudioAnalysis
2) ADB (Android Debug Bridge)
3) avconv

4. System Design

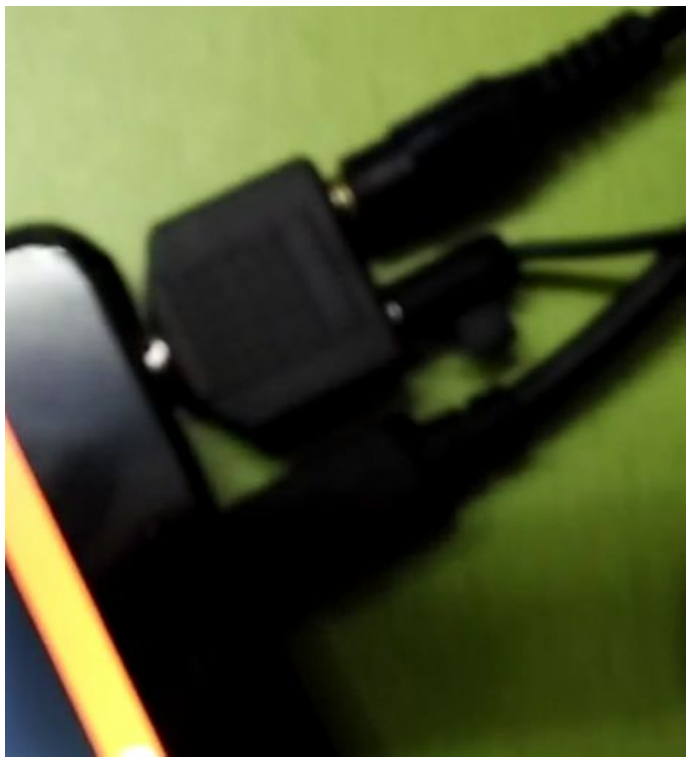


Block diagram

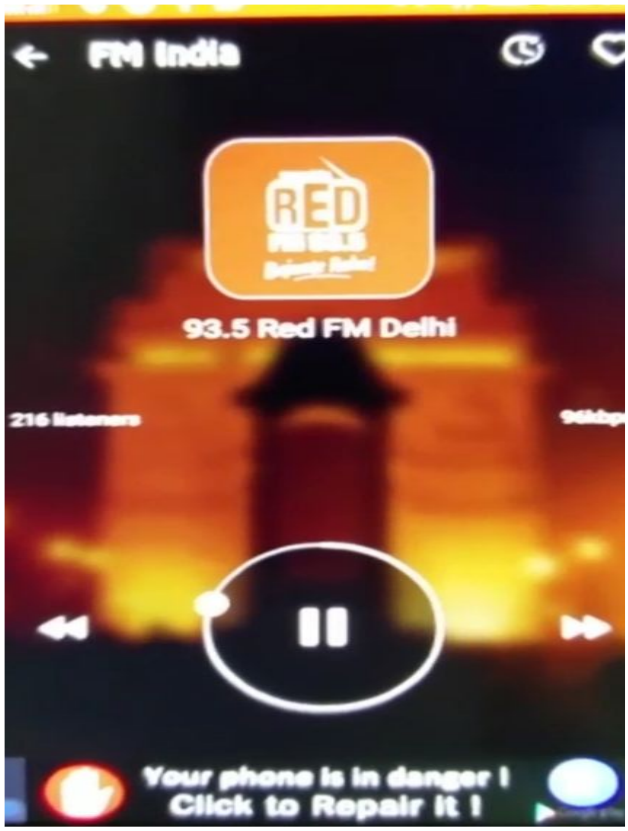
Raspberry Pie with the soundcard



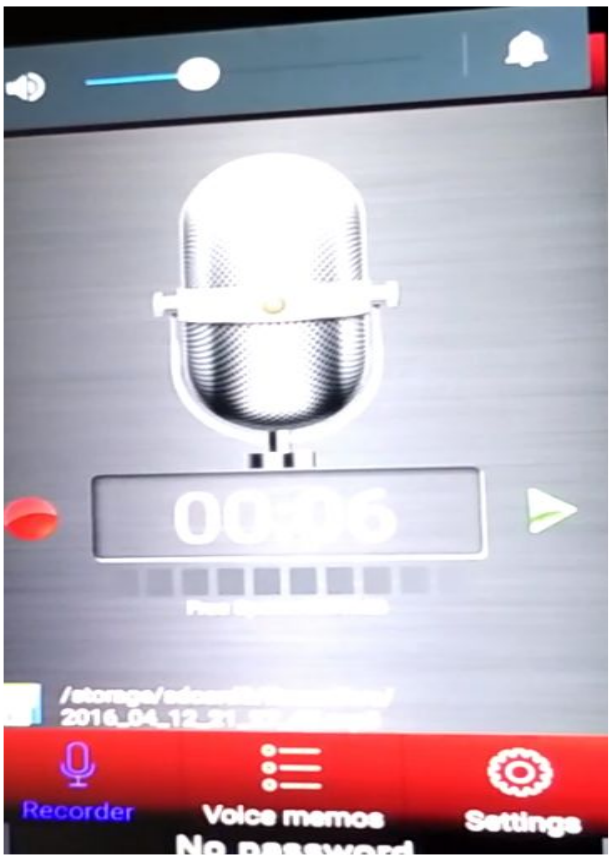
Raspberry Pie connected to the Android device



Audio splitter connected to the android device



Screenshot of mode one



Screenshot of mode two

State Transitions

‘First Mode’ : Changing Radio Frequency

- Normal state: A channel is playing
    - On detecting a human voice signal with probability >= 0.95 => change radio frequency
    - On detecting a human voice signal with probability < 0.95 => stay in the same state

‘Second Mode’ : Volume control based on human voice

- Normal state: Music plays at a given level
    - On receiving an external human voice signal with probability >= 0.5 => go to reduced volume state
- Reduced volume state: Music is playing at a low level
    - On receiving a signal that human voice signal has reduced with probability >= 0.5 => go to normal state

5. Working of the System and Test results

We were very satisfied with the working of the device in both the modes. The testing environment was a controlled environment inside our hostel room. We tested several cases in both 'First Mode': radio frequency changing mode and the 'Second Mode': volume reduction mode.

In the first mode, we had no control over what the radio channel was playing. Yet there was a very good performance in changing the channel when there was only human voice being played/heard.

In the second mode, we tried several songs. While we expected it to work well with instrumental music, which it did. It also worked well with rap music like DJ Bravo's popular 'Champion Song'. In fact in the final demo we showed its successful working to the TAs.

There were almost zero false positives, which should be the aim for such a device. There should not be any annoying/unnecessary changes.

One delay that we incur is owing to the processing time of the relatively weak raspberry-pi processor. On a stronger RAM this delay of 3 seconds that the device incurs to react to a change in environment, would be reduced.

6. Discussion of System

We had planned for two things

1) Changing channel when radio jockey speaks up
2) Changing the music levels according to the conversation intensity levels

We were able to achieve both of the modes but did not implement it in a manner where both the modes can work simulataneously

7. Future Work

Do Not Disturb can be improved at different levels:

1) We can make the product much more compact and pleasant
2) The algorithm implemented does not respond very well to radio jockey with background music, we can work on the machine learning aspect of it and implement a better model
3) The placement of mic to optimally absorb the music and conversation levels is an open ended question and requires some deeper thought
4) Presently, the code has certain amount of latency, we can work on it to reduce the latency

8. Conclusions

Do Not Disturb is an important step towards eradicating human intervention to adjust levels of a system till we find the levels optimal and pleasing. The concept can find some use is taking off the load of changing music levels by the driver. We are pretty sure of existence of varied opportunities in this domain and is yet to be exploited.

9. References

- https://github.com/tyiannak/pyAudioAnalysis
- http://www.csd.uoc.gr/~tziritas/papers/07tmm01-panagiotakis-proof.pdf
- http://www.aclweb.org/anthology/O08-1015