SENSOR BASED VOICE CONTROLLED MUSIC PLAYER

Introduction

In this era of IoT sensors and gesture feature are used in almost every aspects of our life. These sensors are used in order to make our life easier and efficient. Now a days sensors are very common in smartphones and these sensors can be used in the apps to get better user experience. “Music Player” is one of the apps that we use in our daily life. In the modern smartphone there is at least one “music player” app is pre-installed. Apart from that there are lots of “music player” apps and software available in the apps market. Since the beginning of smartphone era the “music player” app have not changed or improved much and the user experience for this have not improved as well. In this experiment we want to show how User Experience (UX) of a “music player” can be improved just by using the built in sensors of the smart phones. For this experiment we used accelerometer sensor, light sensor/ proximity sensor, customized gestures, and voice assistance.

Key Words: IoT, Sensor, Gesture, Android, UX.

Background

Modern smartphones comes with pre-installed music player app. But now a days users use more than one “music player”. Because most of the times users are not satisfied with the experience of the built in music player app. Besides, built in app doesn't give the advanced features that user looks for. Apart from the built in music player app some of the most popular music player app for android devices are [1]: Musicolet, Phonograph Music Player, Pulsar Music player, Pi Music player, BlackPlayer Music Player, n7player Music Player, MediaMonkey, Musixmatch, Netease Music, QQ music, Poweramp etc. Most of these music players are popular because of the UI and UX. For example: “Musicolet” app allows users to control the music player using their earphone button; a single click for pause/play, double click plays the next track, and triple click takes you to the previous song. Also, users can fast-Forward the song with 4 or more repeated clicks. This feature is unique and an ideal example of using “gesture”. Gesture or sensor based apps are always preferred by the users. There are some apps in the app markets that use vibration or in other words accelerometer sensors to simply just change the music. So people prefers these type of apps because are simple and easy to use.

In this report we will point out how a smartphones’ sensors can simply improve the UX of common app.

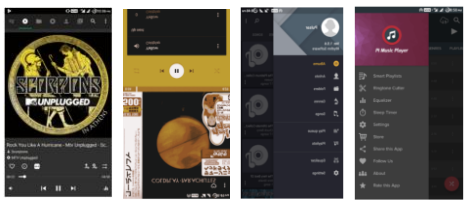

Figure 1: Some popular music player app

VSGMusic Application

We name our android application as “VSGMusic” and it can be described as voice controlled, sensor and gesture based music app. In this part we will discuss about the key features, device requirements and challenges.

Key Features

The key features of the VSGMusic app are as following:

Voice Assistance: Generally in a music player users don’t get the voice assistance app. There are some apps that provides song recognition feature but very few of them has voice assistance in them. So in our app we integrated voice assistance feature.

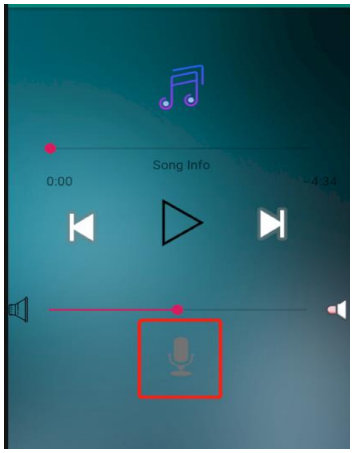

Figure 2: Voice assistance feature

This assistance is not artificially intelligent but it can perform some simple tasks. For example: user can ask it to play current song, play next song, play previous song, stop/pause the current song, also can tell user the current time.

Text-to-Speech: In accordance with the voice assistance we have integrated text-to-speech function. When user gives any voice command via the voice assistance, the text-to-speech feature will speak accordingly. This feature is not so user friendly for a music player but still can be polished so that users get better experience.

Gesture: In this app we have used custom made gesture features. Generally android library provides “tapping”, “double tapping”, “fling” etc. as gesture. But if user wants to do other gestures then it’s not possible to do perform with the traditional gesture library. In this project we backtracked a third party app named “gesture builder” [2]. With the help of this app we custom made three gestures:

1) Tick (✓): plays or pauses the current song.
2) Left to Right (→): plays previous song.
3) Right to Left (←): plays next song.

Actually we can create any gestures and assign that to a task. So after creating the gesture from the described 3rd party app we extracted the raw file from the app by debugging the app. And after that we integrated the gesture in our music player. The gesture extracting process will be described briefly in the upcoming part.

Accelerometer Sensor: We included shaking feature in our app. If the user shake or vibrates the app then it playlist shuffles. We used accelerometer sensor to detect the vibration. The relevant code can be found in “*shakeListener.java”* file. Inside the “*onSensorChanged*” method we sensor values of accelerometer sensor.

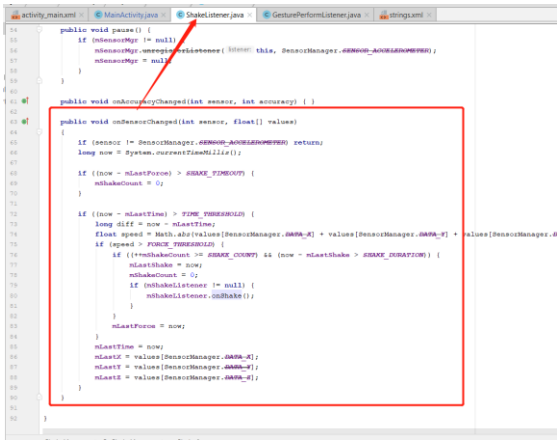

Figure 3: Code for getting Accelerometer Sensor values.

Light / Proximity Sensor: Before playing the song our player check if there is light around. If there is no light the player won't start. Some music players use proximity sensor to detect if any part of user body is near to the device, in that case music player stops. For example: if user place the phone near his ear then the music player will stop or if user flip the phone downwards then music will pause. While testing the features both sensors were giving unstable results. While using the virtual sensors in the emulator sensors worked fine but in an actual device the result was not stable. But still we included this feature and hopefully in future we will be able to develop this feature furthermore.

Stream from Internet: For our player we saved our songs in a cloud service. So after starting the app it starts to stream the song. The original plan was to use the local directory to save the music. But in that case the app size would have be bigger. So taking in consideration of that fact we saved the music/songs in the 3rd party cloud. So every time we play a new song it will take some to load. Therefore the music player take longer time to load any new song which is obviously not efficient.

Device Requirements

To run the app we at least need android Ice Cream Sandwich OS. We also need light sensor, accelerometer sensor, supported gesture library. We also need voice recognition engine for the text-to-speech function. Several permissions are also necessary. Such as: internet permission, read-write the local storage, wake lock permission etc.

```
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

Figure 4: Permissions required for the app

Challenges

The biggest challenge of making this app was to set the gestures. As we mentioned that android library doesn't really support the custom made gesture. To make the custom made gesture we first installed the "gesture builder" android apk in our virtual emulator. After that we run the app. We added our desired gestures.

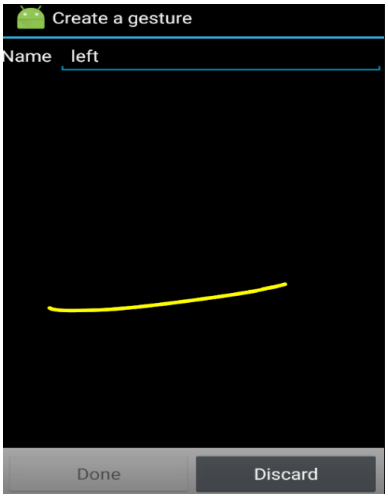

Figure 5: Adding gestures in "Gestures Builder" app

After adding the gestures we debug the app. And find the binary formatted “gesture” saved in a raw file.

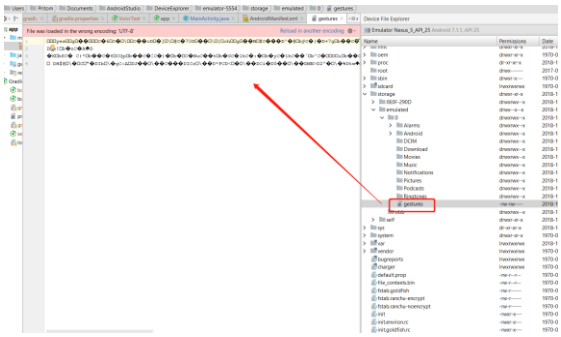

Figure 6: Binary format gesture

We add this file in our raw directory. After that we create a “GestureListener.java” file to do the prediction for the given input. The bottom side of our music played is actually a “gestureoverlay view”. This is where user can give the gesture input.

Figure 7: Gesture input

Apart from the gesture recognition part, setting configuring the voice assistance was also challenging. On the other hand, for shuffling the music we used tricky algorithm. Another challenge was light sensor. Apparently light sensor doesn’t give accurate result in an actual device and also not all the device has the sensor so during the testing in real device the performance was not satisfactory and UX was also quite bad. Therefore we recommend no to use that sensor for music players.

Future Developments

There are bunch of developments can be done in the future. For example the gestures, currently the gestures are hard coded that means users can’t create their own gesture and assign the feature. But in the future it is very possible to introduce this feature. From a user point of view this will be a huge improvement. Currently no music player provides such feature. If we have enough music in our library we can also introduce AI based music preference. Currently lack of data we could not introduce this feature. Since the temperature sensor also doesn’t work well we discarded our original plan of including that. Instead in future we can use 3rd party weather API and create playlist according to the temperature. We also have a plan to use the “accelerometer sensor” and detect the “tilt” of phone and use this behavior to increase or decrease the volume. We have already tested this feature but since it did not give good result we discarded this in our current version.

Conclusion

In this project our vision was to show how the UX of a simple software can be improved by using the existing sensors of the smart phones. We choose music player as our software since we use this every day in our life. Though this app is not life changing but this kind of app is used by almost everyone. And over the past few years, there was no significant amount work for improving this kind of app. We hope our project will give software developers new ideas to improve the software using

gestures and sensors. Hopefully in the future by the improvement of sensors and gestures, we will be able to see brain controlled, retina controlled apps.

Acknowledgement

Finally will thank Professor Yanmin Zhu for conducting the Advanced Topics on Internet of Things class. This class was quite helpful and we learnt many new topics. Discussion on RFID, QR codes, M2M/D2D communication gave us many new information and ideas. Working with Respberry Pi and TinyOS was also quite interesting. Also like to thank teaching assistant who helped us by giving necessary suggestions.

References

[1] [Online]. Available: https://fossbytes.com/best-free-android-music-players/.

[2] [Online]. Available: https://play.google.com/store/apps/details?id=pack.GestureApp&hl=en.