Monitoring App 

In Java with Android Studio

1/02/2020

Giacomo Usai 4390761 giacomo.usai.iic97@gmail.com

Vincenzo Terracciano 4318570 4318570@studenti.unige.it

Introduction

A baby monitor is any tool that records the activity of babies and let parents watch it remotely. Our project is a very simply monitor app that listens the noise and, if the threshold is crossed, notify it. So, in order to develop this project there's a Client phone that is connected to the Server phone.

Hw and Sw Features

We used:

1. **AsyncTask** because this class allows us to perform background operations like monitoring the noise and keep alive the connection between Client and Server.
2. **ServerSocket** class. This class implements server sockets.
3. **Socket** class. This class implements client sockets.
4. **NotificationManager** class for notify the user of events that happen.
5. **MediaRecorder** class is used for record audio. Notice that the function `getAmplitude()` is very important for this project.
6. **Handler class** (https://developer.android.com/reference/android/os/Handler)
7. **SharedPreferences** class for saving (Client side) the last IP and port of the Server.

Monitoring App

This app works using your phone's microphone. You will only need two Android devices. Both devices should have Monitoring App installed. So, via wifi, Monitoring app can connect the Client phone to the Server phone. When the Server hears a noise...it sends a message to the client. So, the Client (that is connected to the Server) will notify the user using a notification like a new Whatsapp message.

How does it work?

This is a tutorial step by step in order to show how to use monitoring app:

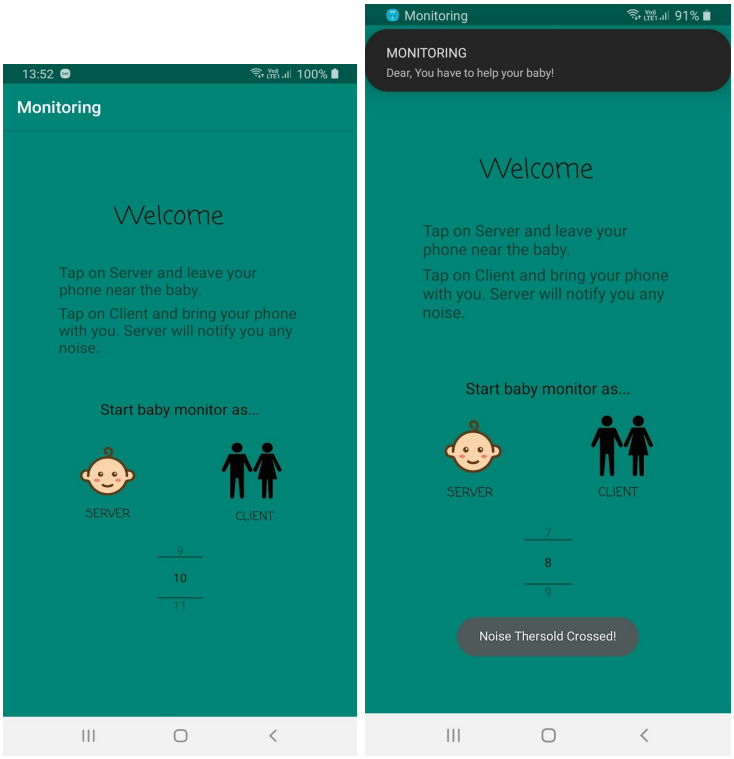
1. Mum's phone and dad's phone are both at home connected to the wifi.
2. Mum opens the Monitoring app and she selects the threshold that she wants via numberpicker selector.
3. Then, she taps on Server (Activity number I and II).
4. Dad opens the Monitoring app, he selects Client and inserts the IP and port of the Mum's phone Server (Activity number I and then number III). Then, he can tap on the green button named "connect ".
    a. If he wants, he can delete all fields with the white button called "clear". Notice that this operation deletes also the information about the last connection (SharedPreferences).
5. So, Mum can leave his phone near the baby and she can go to the bedroom with dad and with the dad's phone.
6. When the Monitoring app running on the mum's phone hears the baby's noise, it sends a message to the dad's phone. So, the dad's phone makes noise (`new NotificationCompat.Builder)(`Activity picture number IV) .

In the same session, if you want to change from Server to Client, you have to close and then re-open the App. There is a special button called "CLOSE" in the main activity.

Notice that, at the end, after that the Server has sent the notification to the client, it waits for a new connection (steps 1,2,3 already gone). In this way, when there's a new connection from the client (step number 4), the Server becomes active.

Activity

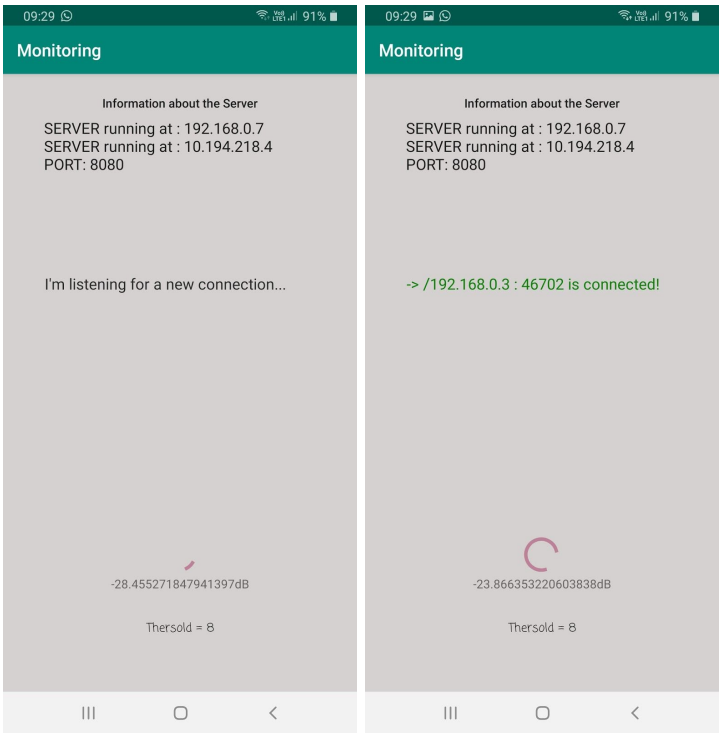
I. MainActivity



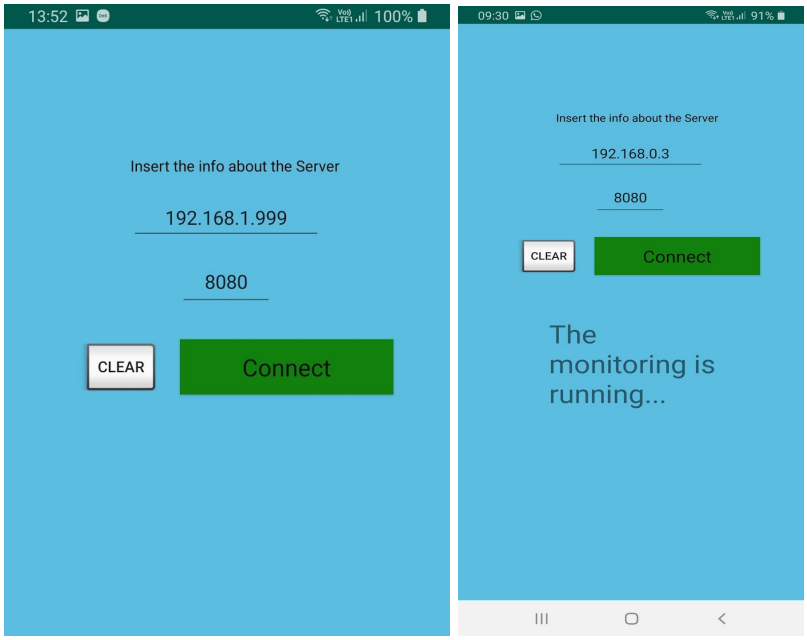
On the left we have the initial homepage.

On the right we have the homepage when the notification is running.

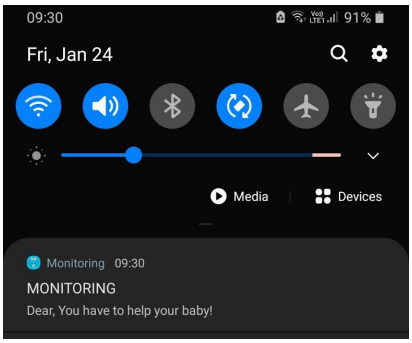
II. Server Side Activity



III. Client Side Activity



IV. Notification



Development information

The main features of this app are two:

1. It can connect two Android devices;
2. It can monitor the noise through the microphone of the device.

The main problem of this project is that our app must to be able to do the main features at the same time. So, in order to solve this problem, we used *AsyncTask* that is used in *Server* and *Client* class because both of them need to maintain the connection alive.

The Server creates a `new ServerSocket(socketServerPORT);` and then `serverSocket.accept();` that allows us to wait until a connection is made.

On the Client side we create a `new Socket(dstAddress, dstPort);` using the information that we should know about the *ServerSocket* created before. Inside the Client class we create an instance of the SharedPreferences class in order to save the last IP and port of the Server.

But how does monitoring app hear any noise?

This process is not easy: the microphone of an Android device is not a sensor such as the devices built-in sensors used in class that measure motion, orientation, and various environmental conditions. There is the MediaRecorder class that is used to record audio and video. The recording control is based on a simple state machine (see https://developer.android.com/reference/android/media/MediaRecorder) and we have to check if the permission for recording audio is granted!

So, we create a `new MediaRecorder();` class and we specify that the audio source must be the microphone of the device. In addition, we set the path to null `myRecorder.setOutputFile("/dev/null");` in order to do not create any file.

Through *getMaxAmplitude()* function (which returns the maximum absolute amplitude that was sampled since the last call to this method) we can understand if the threshold is crossed. When we have to notify that the threshold is crossed, we create a

`new SocketServerReplyThread(socket, msg);`. This class SocketServerReplyThread that extends Thread allows us to create a `new PrintStream(outputstream, true);` and in this way we can send the message to the Client.

When the Client receives a message from the Server, it creates a

`new NotificationChannel("YOUR_CHANNEL_ID","YOUR_CHANNEL_NAME",`

`NotificationManager.IMPORTANCE_HIGH);`.

In this channel we can create a `NotificationCompat.Builder` that creates a typical notification layout such as a WhatsApp message. Finally, we should hear a regular message ringtone.

The main problem for testing this project is that we need two Android device. Android studio can run only one device at the same time!

The Logcat window in Android Studio is very useful because it displays messages that you added to your app with the *Log* class (https://developer.android.com/reference/android/util/Log.html). It displays messages in real time and keeps a history so you can view older messages.

UI information

We created a new logo. The idea was to create the same original Android logo but with a baby instead of the bugdroid (the well known Android mascot).

Textview, button and scrollView are widely used. Moreover, we inserted a numberpicker that allows the user to select the desired threshold. (For the doc. of the numberpicker see https://developer.android.com/reference/android/widget/NumberPicker)

Built with

AndroidStudio
https://developer.android.com/studio/?gclid=EAIaIQobChMIkf6Qzuea5wIVlE8YCh2OMQMdEAAYASAAEgL0NfD_BwE

Tested with

1. Samsung Galaxy Note 9 - Android 10 (API level 29)
2. LG G7 thinQ - Android 9 (API level 28)
3. Xiaomi Redmi Note 6 Pro

Note that you must allow the microphone permission and go into Settings>Notifications>Allow Sound and other.

Possible improvement and features

1. Image surveillance or stream live video: it take a picture of your baby if the alarm is triggered and sent it to another phone.
2. A sleep diary which allows you to take notes and see how long time your baby has been sleeping.
3. When your child wakes, the Baby Monitor will play a song (or mummy's voice) to help soothe them back to sleep.
4. A button that disable and enable the monitoring (microphone) without lose any information.
5. Baby Monitor should work on WiFi and 3G. Use it anywhere without worrying about a weak WiFi signal.
6. If the mum (client) does not re-make the connection after x seconds (so she is near the baby), the app raise up a call.