```
package com.example.listadetarefas;

import android.media.AudioDeviceCallback;
import android.media.AudioDeviceInfo;
import android.os.Bundle;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import android.content.Context;
import android.media.AudioManager;
import android.content.pm.PackageManager;

public void class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      EdgeToEdge.enable(this);
      setContentView(R.layout.activity_doma);
      ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets)
-> {
         Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
         v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
         return insets;
      });
   }
}
public class AudioHelper(context: Context) {
   private val
audioManager:AudioManager=context.getSystemService(Context.AUDIO_SERVICE) as
AudioManager;
   fun audioOutputAvailable(type: int): Boolean {
      if (!
context.packageManager.hasSystemFeature(PackageManager.FEATURE_AUDIO_OUTPU
T)) {
         return false
      })
      return audioManager.getDevices(AudioManager.GET_DEVICES_OUTPUT).any
```

```
{it.type == type}
   })
}
 fun main() {
   private val audioHelper;
   audioHelper = AudioHelper(context.audioOutputAvailable);
   //substiatua context pelo contexto atual do seu dispositivo

   val IsSpeakerAvailable =
audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER); {
    (! if IsSpeakerAvailable =
    return true
       );
   //true if the device has a speaker

   val
isBluetoothHeadsetConnected=audioHelper.audioOutputAvailable(AudioDeviceInfo.TY
PE_BLUETOOTH_A2DP); {
       (! if isBluetoothHeadsetConnected
       return true)
   //TRUE SE O BLUETOOH FONE OUVIDO ESTIVER CONECTADO
}


//|CALLBACK DISPOSITIVO CONECTADO/DESCONECTADO

    audioManager.registerAudioDeviceCallback(object:AudioDeviceCallback) {
   override fun
   onAudioDeviceAdded(addedDevices: Array<out AudioDeviceInfo>?); {
      super.onAudioDeviceRemoved(removedDevices);
      if (! audioOutputAvailable)(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)); {
         return 'Um alto falante acabou de ser conectado'
               else
                  'Um fone de ouvido Bluetooth acabou de ser conectado'
      }
   }
   override fun
   onAudioDeviceRemoved(removedDevices: Array<out AudioDeviceInfo>?) {
      super.onAudioDevicesRemoved(removedDevices);
      if (! audioOutputAvailable)(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)); {
         return 'Um alto falante não está mais conectado'
```

```
            else
                'Um fone de ouvido Bluetooth não esta mais conectado'
        }
    }

    //CALLBACK CONECTADO/DESCONECTADO

    override fun
    onAudioDeviceAdded(addedDevices: Array<out AudioDeviceInfo>?) {
        super.onAudioDeviceRemoved(removedDevices)
        if (! audioOutputAvailable)(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
            return 'Um fone de ouvido Bluetooth acabou de ser conectado'
        }
    }
    override fun
    onAudioDeviceRemoved(removedDevices: Array<out AudioDeviceInfo>?) {
        super.onAudioDevicesRemoved(removedDevices)
        if (! audioOutputAvailable)(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)){
            return 'Um fone de ouvido Bluetooth não está mais conectado'
        }
    }

    //DETECCAO DISPOSITIVO DISPONIVEL

    override fun
    audioOutputAvailable(type: int): boolean {
        if (! )
        //implementacao da funcao audioOutputAvailable

        //retorna verdadeiro se o tipo de dispositivo de audio especificado
        //estiver disponivel.
    }

    //CALLBACK DISPOSITIVO DISPONIVEL

audioManager.registerAudioDeviceAudioDeviceCallBack(object:AudioDeviceCallback) {

    }
```