Digital Health

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF MEDICINE

Digital Health 2021/22
for
Students

Microphone, Loudspeaker and Background Services

by David Kopyto

Date: October 26, 2021
Supervisors: David Kopyto, Dr. Luis Lopera, Dr. Andreas Rowald Prof. Dr. Oliver Amft

1 Introduction

In the last exercise an overview on the cross-platform app development environment Flutter and the programming language Dart was given. The following two sessions aim to give you a hands-on experience with these tools. We will first introduce packages. You will understand how integration of different pre-implemented libraries make Flutter powerful for writing useful apps in a reasonable amount of time. Afterwards, we will discuss some of these packages. During Session 2 audioplayers and microphone packages are addressed. Furthermore, a brief summary of background services will be provided. With this knowledge you will be fully equipped for exercise Session 3, where we gather these parts to build a Sonar app. This app demonstrates how to use the principle of ultrasound-based Radar to detect motions. This idea could be used to build a sleep tracking app.

2 Flutter Packages

Flutter comes with pre-implemented packages which make it possible to access different peripherals, like sensors. These packages either provide platform-specific code for both iOS (Swift, Objective C) or Android (Java, Kotlin) or dart code whenever tasks can be implemented platform independent (usually layout packages etc.). The different packages can be inspected on pub.dev. Installing such a package in your Flutter project is done by updating your pubspec.yaml file and importing it to the .dart file in which you want to use it.

2.1 Installation

To add a package to your project, you first need to open the file pubspec.yaml. Then simply add the package name and the version to your dependencies list. You get this information from the installation instructions on pub.dev for the specific package. For the 'audioplayers" package, which we will also need later, this is shown here:

```
dependencies:
  flutter:
    sdk: flutter
audioplayers: ^0.16.2
```

Next, we need to import the package into our .dart file:

```
import 'package:audioplayers/audioplayers.dart';
```

For some package, it might be also necessary to add some permissions in the android manifest or in the respective iOS file. Such information can be found on pub.dev as well.

3 Loudspeaker

A very important feature of every smartphone is the built-in loudspeaker. In the following, we want to learn how to access it using flutter and the audioplayers package that has already been included to our project in Chapter 2. First, take a look at the documentation at https://pub.dev/documentation/audioplayers/latest/. We now want to initialize an audioplayers object to play a .wav sound file. For that, we write a new function:

```
void initPlayer(){
  advancedPlayer = new AudioPlayer();
  audioCache = new AudioCache(fixedPlayer: advancedPlayer);
}
```

This function initPlayer() initializes the audioplayer for the moment when the app is started. An AudioCache object connected to the AudioPlayer also must be created. We will later need to load the audio file which we want to play in this cache. Before that, we need to make sure that the audio file we want to play is correctly placed into our project. This is done by adding assets to it.

3.1 Adding Assets to the project and Loading them to AudioCache

We now want to include audio files to our project. The first thing we need to do is to add a folder 'assets' to our directory. Afterwards, the pubspec.yaml file needs to be modified as follows:

```
flutter:
  assets:
    - assets/
```

This includes assets in the folder to the project. We can now put .wav files to the assets folder and call them in our code. This can be done by the following:

```
audioCache.loop('20kHz_test.wav'); // infinite loop of the audio
audioCache.play('20kHz_test.wav'); // play audio once
```

The player can be stopped by:

```
advancedPlayer.stop();
```

4 Microphone

The overall goal of these exercises is to design a Sonar. A Sonar is an acoustic based Radar system, which aims to track motions using the microphone and the loudspeaker.

The principle of this system is discussed in further detail in Session 3. To set this system up, we now need to access the built-in microphone. This is done using the package 'record'. Please look at: https://pub.dev/documentation/record/latest/. In the following we set up the microphone in our app. Before using the microphone, we need to ask for permission to the user.

```
bool result = await Record.hasPermission();
```

As you might notice, the function call includes the 'await' keyword. This means the function is called asynchronously, which needs to be kept in mind when using it

```
await Record.start(
  path: wavPath, // required
  encoder: AudioEncoder.AAC, // by default
  bitRate: 128000, // by default
  samplingRate: 44100, // by default
);
```

4.1 Storing the Recorded File

Let us take a quick look at the parameters we have set for the microphone. We need to first set a path wavPath. This string contains the path to where the data from the microphone is stored. For that, we need to set a global variable in which we write this path beforehand. This is done using the path_provider package from flutter (for further information please check: https://pub.dev/documentation/path_provider/latest/). With this package we can access the internal directories of the phone in which we can store data. We do this by:

```
io.Directory tempDir = await getExternalStorageDirectory();
```

The result is a directory file (from dart:io library). We add an audio file to this directory, convert it to a String and save it to the variable we just included:

```
io.File tempFile = io.File('${tempDir.path}/audio_data.m4a');
wavPath = tempFile.path.toString();
```

This enables us to store data from our microphone. When we want to stop the recording, we use

```
await Record.stop();
```

5 Background Services

To ensure that microphone and loudspeaker do not cease to run when the app is running in the background, Flutter provides some workarounds. To take a further look at that, please check https://pub.dev/documentation/flutter_background_service/latest/ and https://pub.dev/packages/android_alarm_manager . Background services are usually platform specific. That is due to different restrictions on Android and iOS respectively. Fortunately, packages such as audioplayers already provide implementations which both work on iOS and Android in the background. The great strength of Flutter is now that it will integrate both implementations to the project and calls the right one depending on where the app is installed. For that, Flutter makes it possible to run on both platform with the same project. If you want to run your own routines in the background, you will need to write these specifically for Android (Java, Kotlin) or iOS (Swift, Objective C).

6 Questions and Exercises

1. Packages: Browse through pub.dev and look specifically at the packages audioplayers, recorder and scidart. Include these three packages to your project. Summarize their usage quickly.
2. Loudspeaker: Include these three packages to your project.
3. Loudspeaker: Include the assets folder to the project.
4. Loudspeaker: The widget "FloatingActionButton" is starting the procedure of the app. Include a function which calls the loudspeaker such that this button plays and stops the audio file. 20kHz_test.wav when it is pressed. Summarize how you preceded in the report.
5. Microphone: Provide the variable wavPath and assign the path in the code. In which other ways can the smartphone memory be accessed? Give a summary.
6. Microphone: Ask for permission for using the microphone to the user in your code. Which keyword do you need to make this function call?
7. Microphone: Start and stop the microphone at the appropriate position in the code. We want to enable/disable both the microphone and the loudspeaker using the same button. Summarize how you preceded in the report.
8. Background Services: How do iOS and Android differ in the restrictions for backgrounds services? Give a summary of what you find on the Internet.