USER GUIDE

How to install and use the My Brain Technologies’ SDK in your projects

# General Overview

This document aims at helping users to install and use the My Brain Technologies’ SDK in their own Android project. It is arranged in a way to guide you through the various steps in your development process. We recommend following each of the steps outlined below and reading the documentation in the order displayed.

The SDK is a closed source library developed by My Brain Technologies. Its main purpose is to allow the development of external applications based on My Brain Technologies’ headset. It also provides code samples, and technical and use case documentation for developers.

My Brain Technologies’ headset includes Melomind and VPro headsets:

* Melomind is an audio headset composed of 2 electrodes that record electrical activity of the brain. These 2 channels of EEG acquisition are located on P3 and P4 parietal position.
* VPro is a headset composed of 8 electrodes that record electrical activity of the brain. These channels of EEG acquisition can be located at customized position of the brain

This library is currently compiled as a .aar file. Its content is obfuscated. Only the public content is accessible to external applications.

# Versions and features

## Requirements & configurations

Using the My Brain Technologies’ SDK requires to install an IDE for developing Android applications. *Warning: this document explains how to install the SDK on Android Studio IDE only.*

The My Brain Technologies’ SDK is compiled with Android API version 27 and is compatible with Android API version 22 and higher. Older versions will not work.

Please adjust your application’s minimum API version accordingly.

## Features

The current version of the SDK is 2.0.0. The SDK supports the following features:

Bluetooth Features:

* Bluetooth Low Energy connection with Melomind headset
* Bluetooth Low Energy disconnection with Melomind headset.
* Bluetooth Serial Port Profile connection with the VPro headset.
* Bluetooth Serial Port Profile disconnection with the VPro headset.
* Starting streaming for receiving EEG data acquired by the headset in the application
* Stopping streaming for not interrupting EEG data reception

EEG Data Processing Features:

* Processing of EEG raw data acquired by the headset including
  + Conversion of EEG raw data into user-readable EEG data values.
  + Notification from SDK to User Application that return the EEG converted values.
  + Computation of the resulting status corresponding to an EEG data.
  + Computation of the signal quality for each channel of EEG acquisition of the headset.
  + Computation of the resulting relaxation level corresponding to an EEG data.

EEG Data Recording Features:

* Recording of EEG data acquired by the headset including
  + Starting a new recording that contains user-readable EEG data.
  + Stopping a current EEG data recording
  + Storage of the data recordings into JSON files.
  + Sending of the JSON files that contains EEG data to the server.

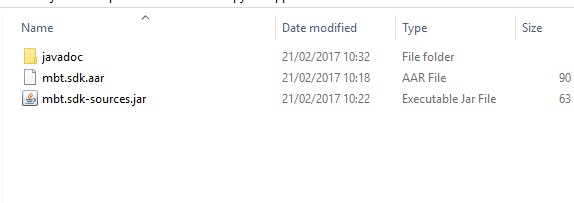
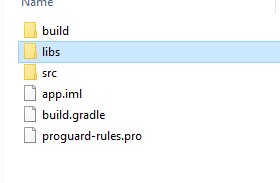
Further updates will be released with more content.

# How to install the SDK on your project

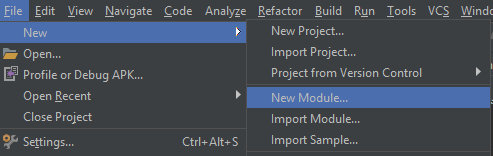
1. Create a new Android Application Project
2. Add the following permissions to your AndroidManifest.xml CHECK THIS

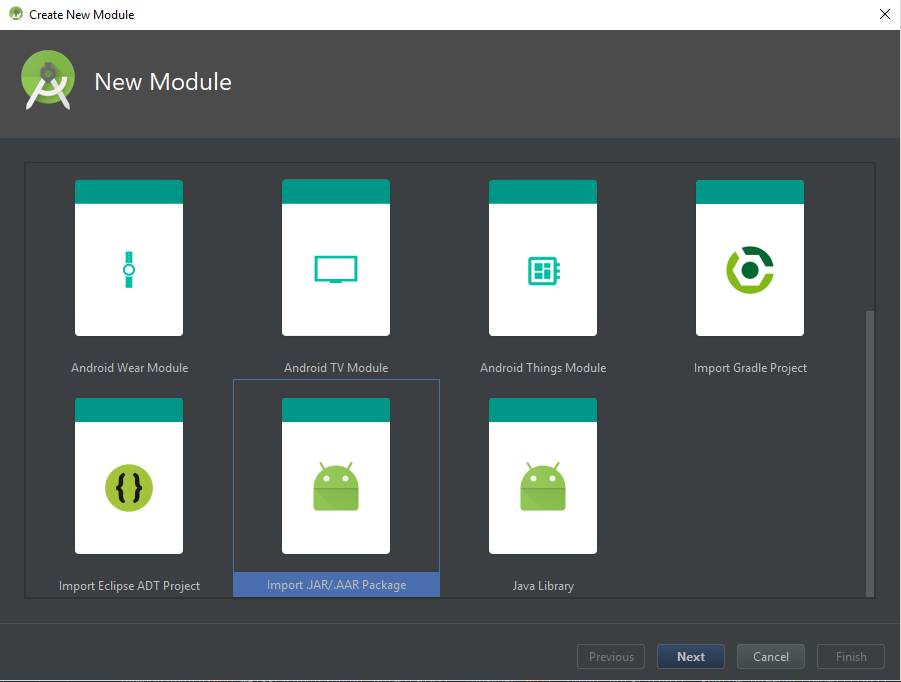
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
<uses-permission android:name="android.permission.VIBRATE"/>  
<uses-permission android:name="android.permission.INTERNET" />  
  
<permission android:name="android.permission.MANAGE\_DOCUMENTS" />  
<permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
<permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

1. Inside the “app” folder, create a “libs” folder and copy paste the javadoc folder, the .aar library and the .jar sources.

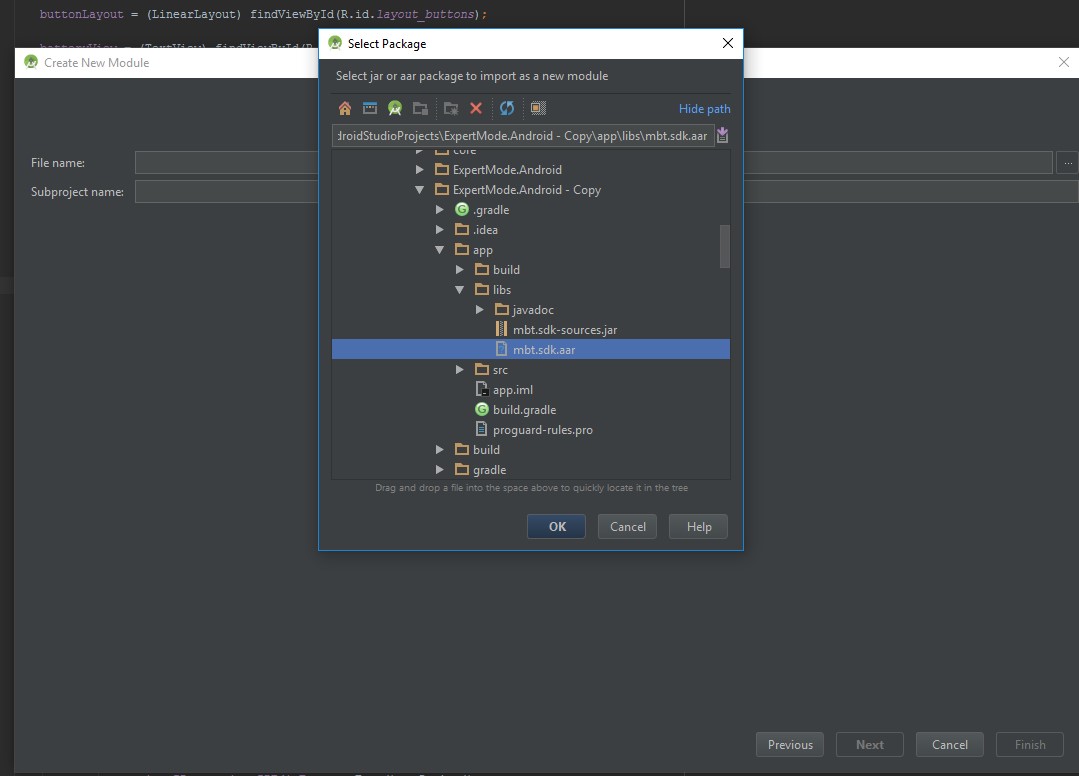


1. Add a new module in your Android Studio project as .aar package. Go to File > New > New Module. Then scroll down and choose “Import JAR/AAR Package”.

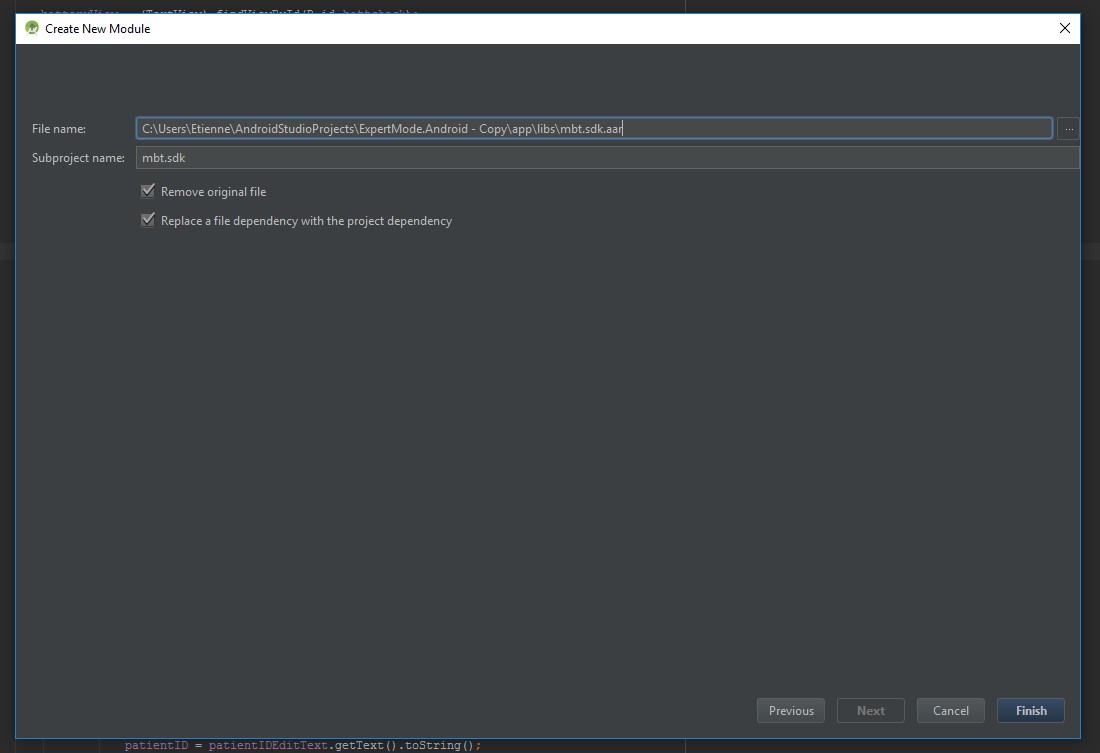




1. Click Next, then select the .aar library previously added in the libs folder.



1. Press OK, then Finish. Keep the library name as mbtsdk.



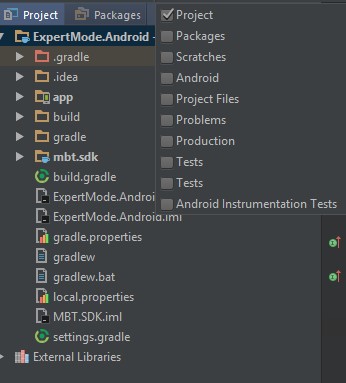
1. Now the library is imported. A mbtsdk project should be visible. Make sure the library has been added to the project dependencies of your gradle file as follow :



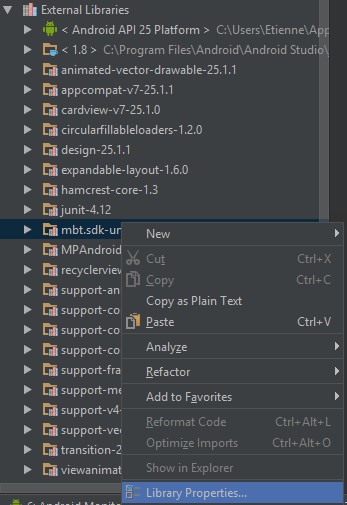
Click on Sync Now in the upper right of Android Studio if a yellow notification bar appears or go to File > Sync Project with Gradle Files.

Your project can use it but all the code is obfuscated. In order to have access to non obfuscated code and javadoc, it is necessary to add both manually as Gradle don’t support automatic integration yet.

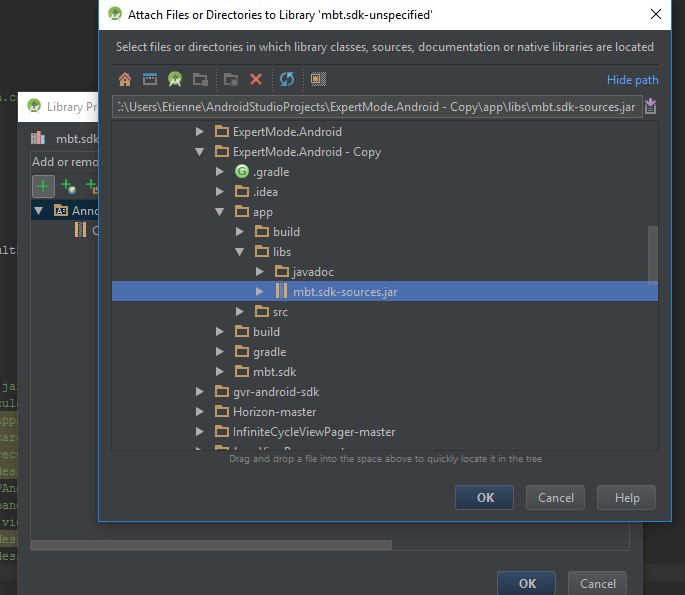
1. To add sources and javadoc to the library, first switch to “Project” view.



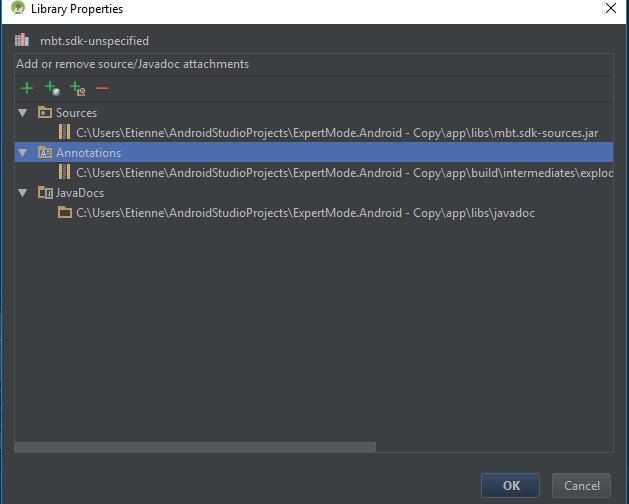
1. Then open the “External Libraries” folder and look for “mbtsdk”. Version might be unspecified. It is not a problem. Then right-click on the library and click “Library Properties”



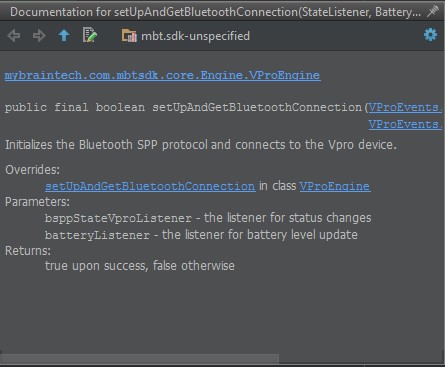
1. Add new properties by clicking on the top left green plus. Look for the .jar and the javadoc you previoulsy added in the “libs” folder. Select them and Click OK.



1. The Proprties Javadoc and Sources have been added to the properties list.



1. Press OK. Javadoc is now available. Obfuscated code in the .aar library will appear in red. It is normal. Everything should run fine.



# How to use the SDK

## Communication between SDK & application

Once the SDK is installed on the application project, the users can use the several methods for developing their own application. The SDK communicates with the application through a client: the MbtClient.

The first step is to create a MbtClient instance and initialize this client inside the OnCreate method of the main activity of the application.

private MbtClient client;

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);

MbtClientEvents mbtClientEvents = new MbtClientEvents();client = MbtClient.*init*(getApplicationContext(),mbtClientEvents);  
}

When a MbtClient instance is initialized, it constructs a MbtManager object, that provides the user access to the Bluetooth and EEG processing methods.

The application is now set up to interact with the SDK features.

## Bluetooth Features

The Bluetooth communication between the headset and the application is managed by the MbtBluetoothManager. This object allows Bluetooth devices detection, connection, disconnection, data streaming, reading informations about the device (headset) and configuring the headset.

client.scanDevicesForType(deviceType, duration, scanCallback)

client.connectBluetooth()

client.disconnectBluetooth()

client.configureHeadset()

client.readBattery()

client.stopReadBattery()

client. startstream(useQualities, clientEvents)

client.stopstream()

1. EEG Features

The EEG data processing is managed by The MbtEEGManager. This object allows conversion of EEG data acquired into user-readable values, computation of the EEG signal quality and computation of the relaxation indexes.

For using the conversion feature for a single raw EEG data acquired, the user needs to call the following method:

client.launchConversionToEEG(data, protocol)

Parameters

@NonNull byte[] data is a non null byte array that contains the EEG raw data.

BtProtocol protocol data is the Bluetooth protocol used to transmit EEG data from the headset to the application (Low Energy or Serial Port Profile).

For using the conversion feature for a continuous acquisition of data, the user needs to call the following method:

client.handleDataAcquired(data)

Parameters

@NonNull byte[] data is a non null byte array that contains the EEG raw data.

For using the EEG signal quality computation feature, the user needs to call the following method:

client.computeEEGSignalQuality(sampRate, packetLength, channels)

Parameters

int sampRate is the sampling rate

int packetLength is the length of a EEG packet(=time x sampRate)

Float[] channels is the array that contains the EEG channels of acquisition

For using the EEG relaxation index computation feature, the user needs to call the following method:

client. computeRelaxIndex(sampRate, calibParams, packets)

Parameters

int sampRate is the sampling rate

MbtCalibrationParameters calibParams is the object that contains the calibration parameters previously performed

MbtEEGPacket packets is the object that contains EEG data, theirs status and qualities.

1. Recording Features

The EEG data recording is managed by The MbtRecordingSessionManager. This object allows recording the EEG data acquired : a recording is used to save these data into internal JSON files, and sending these JSON files to the server for an external save in the database.

1. Server Features

