

**CALM. SDK API Manual**

**(android)**

Table of Contents

[1. Introduction 2](#_Toc488923886)

[1.1 Purpose 2](#_Toc488923887)

[1.2 Operating environment 2](#_Toc488923888)

[2. SDK Summary 3](#_Toc488923889)

[3. SDK Access Description 4](#_Toc488923890)

[3.1 Integration 4](#_Toc488923891)

[3.2 Interface Instruction 5](#_Toc488923892)

[3.3.1 Main Functions 5](#_Toc488923893)

[3.3.2 Callbacks 5](#_Toc488923894)

[4. Demo Explanation 7](#_Toc488923895)

# **1. Introduction**

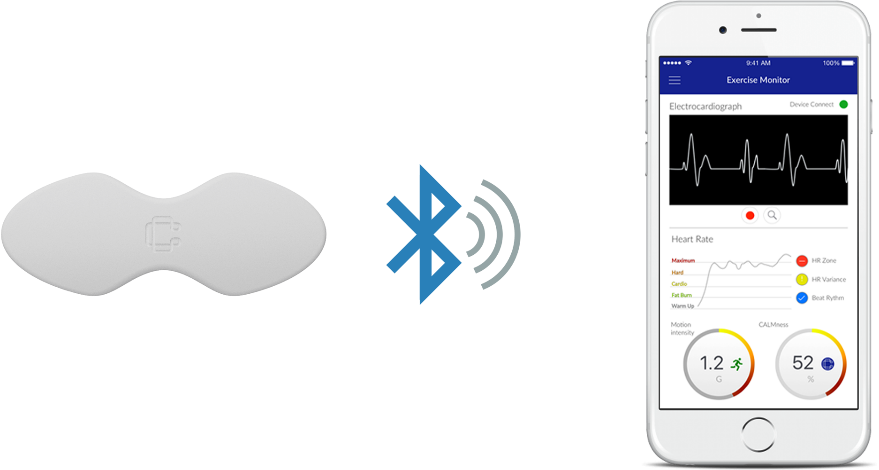
## **1.1 Purpose**

This manual explains how to use CALM.SDK to develop custom mobile app which interacts between CALM. device and mobile phones.

**Prerequisite:**

* Android 4.3 or above
* CALM. Device
* Android Studio v2.0.0 or above

## **1.2 Operating environment**

****

**Figure 1.1 Operating Environment**

# **2. SDK Summary**

CALM.SDK has 4 classes exposed to developers. (Details will be explained later)

1. CALM

Main class of CALM.SDK which expose main functions of the sdk.

You have to initiate CALM object to get started with CALM. SDK.

1. CALMCallback

Callback interface for CALM main functions.

Ex) when you call Scan() function on CALM object, then onScanFinished() callback will be called with the scan results.

1. CALMData

Used for transferring data between SDK and user app.

Data will contain ecg, acceleration, batter level, temperature.

1. CALMStatus

Used for transferring status between SDK and user app.

Data will contain connection status, mac, rssi, firmware version.

# **3. SDK Access Description**

## **3.1 Integration**

SDK is distributed as Android Library(calm.sdk.aar)

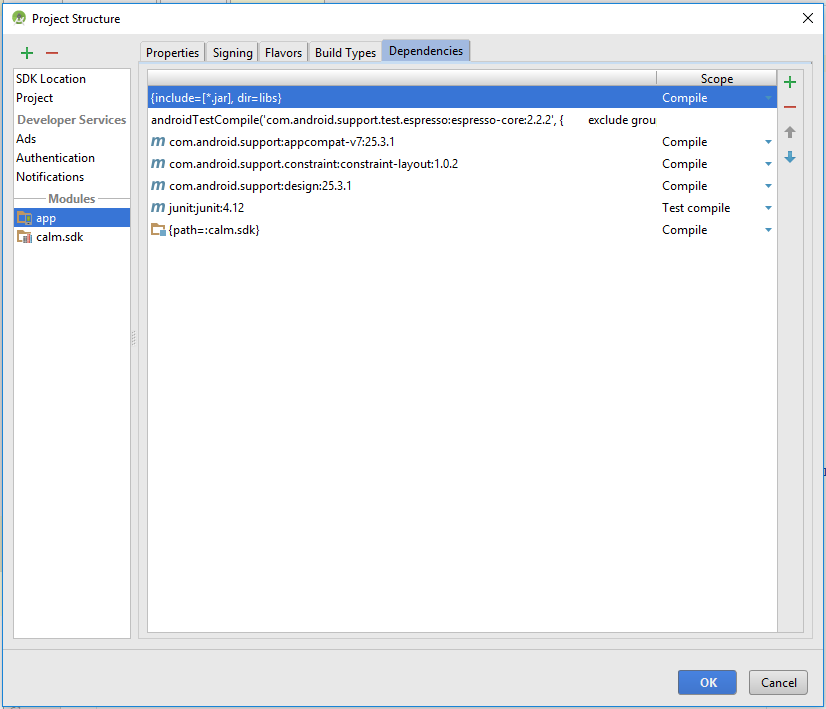
1. Create or open existing Android project
2. Open “Create New Module” by going on menu – File/New/New Module

Select Import .JAR/.ARR Package then choose location of sdk file.

Then you have now imported library to your project.

1. Make the library as dependency.

Go to Project Structure (File/Project Structure – Ctrl + Alt + Shift + S).



On the left pane, there is two modules – app, calm.sdk

Select app, then go to “Dependencies” tab on the right pane.

Click the green add button on the top right, then select “Module Dependency” from dropdown menu.

From the popup, choose “calm.sdk” module.

Congrats!!! Now you have successfully integrated CALM. SDK to your project.

## **3.2 Interface Instruction**

### **3.3.1 Main Functions**

Class name – CALM

1. Initialization of CALM

CALM(Context context, CALMCallback callback)

@context – Context in which CALM is initiated.

@callback – Callback object for main functions

Ex) CALM mCalm = new CALM(this, mCalmCallback)

1. Scan(int milliseconds)

Scan for CALM. Devices.

You can set milliseconds for scanning duration, which default is 5 when millisecond is not defined.

When scan is finished, onScanFinished() callback will be invoked with scan results.

Ex) mCalm.Scan();

1. Connect(string mac)

Connect to CALM. Device by mac address, which you acquired from Scan() function.

onConnect() callback will be invoked with result whether connection success or failed.

Ex) mCalm.Connect(“AA:BB:CC:DD:EE:FF”);

1. Disconnect()

Disconnect any connected CALM. Device

onDisconnect() callback will be invoked when disconnected.

Ex) mCalm.Disconnect();

1. GetStatus()

Returns status of CALM.device.

onCALMStatusRead() callback will be invoked once read status.

Ex) mCalm.GetStatus()

When device is connected successfully, then sdk will invoke onCALMDataRead() callback whenever it gets new data from device.

### **3.3.2 Callbacks**

Class – CALMCallback

1. void onScanFinished(List<BluetoothDevice> foundDevices)

Invoked when scan has finished.

1. void onConnect(Boolean success)

Invoked when connection finished with result of connection.

1. Void onDisconnect()

Invoked when CALM. Device disconnected.

1. Void onCALMDataRead(CALMData data)

Invoked when received new data from CALM. Device

1. Void onCALMStatusRead(CALMStatus status)

Invoked when status of device is read

Next, you should be familiar with CALMData, CALMStatus before getting started with callbacks, because they are basic data structure for transferring data, status.

1. CALMData

* boolean isBodyDetected();

Returns status if user put on or off the device.

* ArrayList<Integer> getEcgArray()

Returns ECG values as array.

* int getTemperature()

Returns temperature of device

* int getBattery()

Returns battery level of device

* Acceleration getAcceleration()

Returns acceleration data

class Acceleration {

public int X;

public int Y;

public int Z;

}

1. CALMStatus

* boolean isConnected()

Returns if devices is connected to phone.

* String Address()

Returns macAddress of connected device

* int RSSI()

Returns rssi of device

* FWVer()

Returns firmware version of connected device.

# **4. Demo Explanation**

The demo app demonstrates usage of SDK to communicate between CALM. Device and mobile phone via BLE.

Functionalities of app is – Scan, Connect, Show ECG Graph in real time, Show device status

First create CALM, CALMCallback objects inside MainActivity

-------------------------------------

**private** CALM **mCalm**;  
**private** CALMCallback **mCALMCallback**;

--------------------------------------

Create initializeCALM() function to initiate mCallback, mCalm then call it on onCreate()

If you need to do UI touch inside callbacks, then you will have to run code on runOnUiThread(), because callbacks are not invoked from main thread.

--------------------------------------

**private void** initializeCALM() {  
 **mCALMCallback** = **new** CALMCallback() {  
 **@Override  
 public void** onScanFinished(List<BluetoothDevice> foundDevices) {  
 **for**(**int** i = 0 ; i < foundDevices.size() ; i ++) {

String newDeviceAddress = foundDevices.get(i).getAddress();

}  
  
 runOnUiThread(**new** Runnable() {  
 **@Override  
 public void** run() {  
 {

**// Do UI stuff here**  
 }  
 }  
 });  
 }  
  
 **@Override  
 public void** onCALMDataRead(**final** CALMData data) {  
 **int battery =** data.getBattery();  
 **int temperature =** data.getTemperature();  
 **int accelX =** data.getAcceleration().**X;**

**int accelY =** data.getAcceleration().**Y;**

**int accelZ =** data.getAcceleration().**Z;**  
 **boolean bodyDetected =** data.isBodyDetected()  
 ArrayList<Integer> ecgArray = data.getEcgArray();  
 runOnUiThread(**new** Runnable() {  
 **@Override  
 public void** run() {  
  
 {

**// Do UI stuff here**  
 }  
 }  
 });  
 }  
  
 **@Override  
 public void** onCALMStatusRead(**final** CALMStatus status) {  
 runOnUiThread(**new** Runnable() {  
 **@Override  
 public void** run() {  
 {  
 **tx\_rssi**.setText(**"RSSI: "** + status.RSSI() + **" dB"**);  
 **tx\_fwver**.setText(**"Firmware Ver: "** + status.FWVer());  
 }  
 }  
 });  
 }  
  
 **@Override  
 public void** onConnect(**final boolean** success) {  
 runOnUiThread(**new** Runnable() {  
 **@Override  
 public void** run() {  
 {  
 **if**(success) {

} **else** {

}  
 }  
 }  
 });  
 }  
 **@Override  
 public void** onDisconnect() {  
 runOnUiThread(**new** Runnable() {  
 **@Override  
 public void** run() {  
 { }  
 }  
 });  
 }  
 };  
 **mCalm** = **new** CALM(**this**, **mCALMCallback**);  
}

--------------------------------------

Now you can call main functions using CALM object.

mCalm.Scan();

mCalm.Connect();

mCalm.Disconnect();

mCalm.GetStatus();