



Wearable Devices LTD

Mudra API and SDK

October 2017



Release Notes

Author: Guy Wagner

Title: VP R&D

Version: 01

Date: 18/10/2017

Table of Contents

Version Release Notes	2
Table of Contents.....	3
1. Device Overview.....	5
a. Device Gestures	5
b. Device Hardware	5
c. Current Use-Cases	5
2. Android Application.....	6
a. System Requirements.....	6
b. Application Screens	6
3. User-Guide.....	7
a. Charging the Device	7
b. Downloading the Application.....	7
c. Wearing and Sizing.....	7
d. Pairing the Device.....	7
e. Streaming Live Signals	7
f. Calibration Procedure	8
g. Diagnostics and Verification.....	8
h. Power and LED Indicator Guide.....	8
4. API - Android	9
a. Implementing the API.....	9
b. Android APIs.....	9
c. IMudraAPI.aidl Interface	10
d. IMudraDeviceStatusListener.aidl Interface.....	15
e. IMudraDataListener.aidl Interface	17
5. API - Unity.....	19
6. API – Android Wear.....	20
7. Use-Case Applications	21
a. Mudra AR App	21
b. Mudra VR App	22
c. Mudra Smartwatch App	23
8. FAQ and Known Issues	24
9. Contact Details	25
Sources.....	26

Applications	26
Documentation	26

1. Device Overview

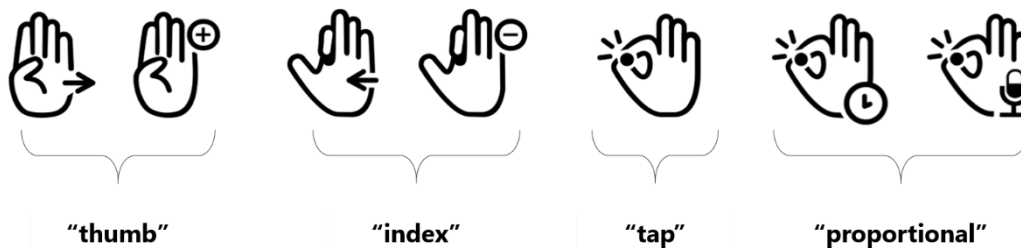
Mudra – Sanskrit (मुद्रा) for "hand gesture" – is a wrist-worn wearable controller. When the user moves his fingers, sensors mounted on the device detect biometric signals originating from the nervous system passing through the wrist. Deep learning algorithms translate the signals and machine learning algorithms classify them as a user-intended gesture. Each gesture defines a unique control operation on the device.

Once a gesture is identified and classified, it is transmitted to the device's operating system, where each gesture is associated with an operation on the device – select, drag & drop, scroll, etc.

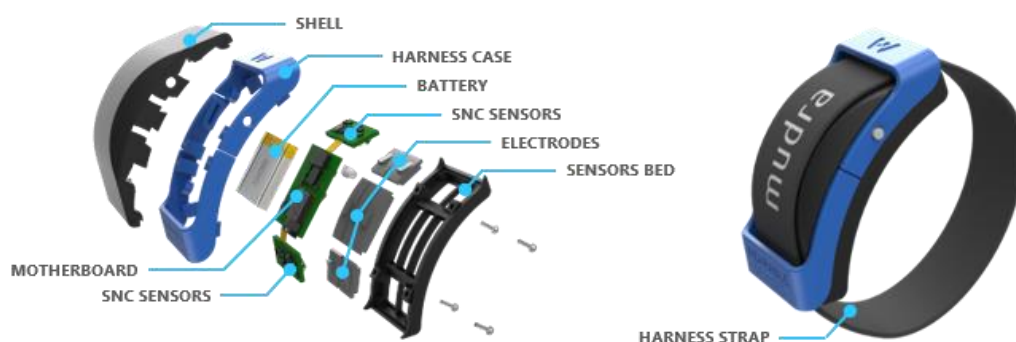
a. Device Gestures

The current device recognizes three discrete gestures and one continuous gesture:

- the individual movement of the thumb finger, a.k.a. "thumb"
- the individual movement of the index finger, a.k.a. "index"
- the soft tap of the index finger with the thumb, a.k.a. "tap"
- gradations of continuous pressure index finger applied on the thumb, a.k.a. "proportional"



b. Device Hardware



c. Current Use-Cases

- Smartwatch control – using "thumb" and "index" to swipe up/down or left right, "tap" to select, and "proportional" for continuous command such as recording a voice message
- AR/VR headset control – using "continuous" for drag & drop, "tap" to point/select objects, "index" and "thumb" to navigate through menus, and apply varying pressure on objects

2. Android Application

The Mudra app is the main engine that will allow you to control your devices using our custom SNC sensors.

The app runs a service on a smartphone which receives raw data from the Mudra capsule over BLE. Using these sensor readings, the app extracts gestures and proportional pressure applied by the user and communicates such events to other apps via the Mudra API.

The Mudra App is the front panel that searches and connects your phone to the Mudra device, allows gesture calibration, viewing raw signals and detects gestures/pressure.

a. System Requirements

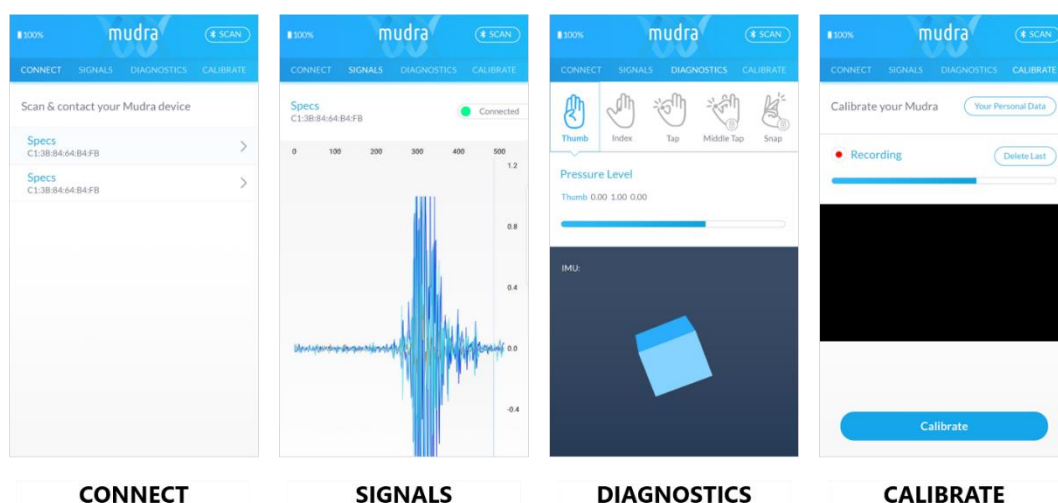
- Android smartphone version 6 or later
- Snapdragon 808/650 equivalent or better with at least ARM-V7 CPU
- 2GB Ram

b. Application Screens

The application has four main screens – Connect, Signals, Diagnostics, Calibrate. You can move between screens by tapping on a screen title name on the top navigation bar, or by swiping to the left and to the right.

The screens' functionality is:

- **CONNECT** – pairs the device to the smartphone via BLE
- **SIGNALS** – displays live signal stream from each SNC sensor
- **DIAGNOSTICS** – displays the gesture the user performed
- **CALIBRATE** – builds a unique model for each user's physiology



3. User-Guide

After unboxing the device, please follow these guidelines and complete an initial calibration procedure for each device user. You may also refer to the Capsule Brochure document in the Appendix for additional guide.

a. Charging the Device

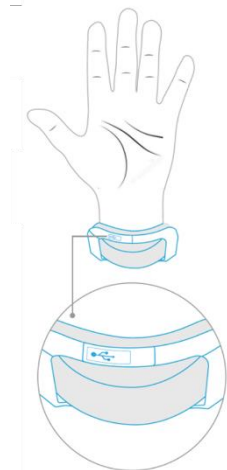
- Open the USB charging socket in Mudra capsule
- Connect the USB cable plug into the socket
- First time charging should be at least 1 hour before use
- Led indicator light is solid orange while capsule is charging

b. Downloading the Application

- Download the 'Mudra Capsule' application from the Google Store
- Install the application on a mobile phone

c. Wearing and Sizing

- Place the Mudra capsule on your left-hand wrist
- Verify that the USB charging socket is on the top
- Tighten the strap to attach the capsule to your wrist at your convenience



d. Pairing the Device

- Open the Mudra application
- Click the power button on the capsule to turn the device on; Led indicator light is pulsing green
- On the CONNECT screen, tap the SCAN button of the top right
- Tap to choose WLD Mudra, which is the capsule device
- The message 'Device Connected' appears when the capsule is paired
- Led indicator light is solid green

e. Streaming Live Signals

- Tap 'SIGNALS' on the top navigation bar
- Gently move each finger
- The live stream displays three signals, one for each device sensor; notice how each finger movement creates a unique signal pattern on the device

f. Calibration Procedure

- Tap 'CALIBRATE' on the top navigation bar
- Tap 'User Data' and fill user details
- Tap 'Start Record' to initiate a new calibration
- The calibration procedure requires 15 movements of the thumb finger, followed by 15 tap gestures, and finally 15 index finger movements
- Calibration progress is visible on the top blue bar
- When the calibration is finished, tap the Calibrate button to apply the new calibration

g. Diagnostics and Verification

- Tap 'DIAGNOSTICS' on the top navigation bar
- Make a tap, index or thumb gesture; Verify the device correctly recognizes the gesture
- Move your hand on the X, Y and Z axes and watch the IMU orientation, which is visualized at the bottom of the screen






h. Power and LED Indicator Guide

The device uses a lithium battery. Charging the device requires a USB to micro-USB cable (supplied).

The device has a single power/reset button, and a LED indicator.

A short press on the power button will perform a SW reset. A long press (5 sec) will perform a HW reset which refreshes connectivity info.

The following LED indicators provide info about the state the device:

-  Charging (solid orange)
-  Low battery connection (fast pulsing red)
-  Low battery transmission (slow pulsing red)
-  Device is paired (fast pulsing green)
-  Looking for pairing (slow pulsing green)

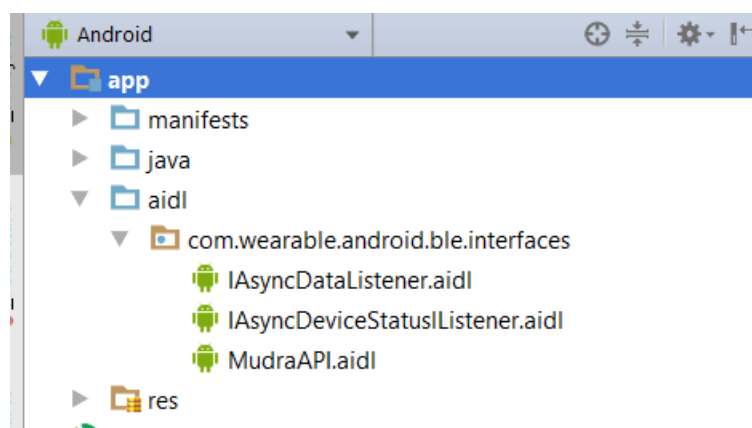
4. API - Android

Android can be used for creating new experiences, such as VR experiences using an HMD or AR experiences without such an addon. For such purposes, we utilize AIDL as an interface, see [AIDL](#) for generic info.

a. Implementing the API

To start working with the Mudra API in android, follow these steps:

1. Create AIDL directory `app\src\main\aidl\com\wearable\android\ble\interfaces` and copy the AIDL files to it. Your directory tree should look like this:



2. Import the AIDL files in the MainActivity.java
 - `import com.wearable.android.ble.interfaces.IAsyncDataListener;`
 - `import com.wearable.android.ble.interfaces.IAsyncDeviceStatusListener;`
 - `import com.wearable.android.ble.interfaces.MudraAPI;`
3. Bind to the mudra service and initialize (see `initMudra`)
4. Create callback function to handle incoming gestures (see `MudraDeviceStatusListener`)
5. Create callback function to handle Mudra status (see `IMudraDataListener`)
6. Call API functions to manage your Mudra device (see `IMudraAPI`)

b. Android APIs

The following sections describe each Interface' core functions, parameters, code blocks and examples. The description includes Member functions, Parameters, Usage examples. The APIs are:

- [*IMudraAPI.aidl Interface*](#) – managing connection / pairing directly with the device, without the need to open the Mudra Capsule application
- [*IMudraDeviceStatusListener.aidl Interface*](#) – receiving connection / pairing status of the capsule with the paired device
- [*IMudraDataListener.aidl Interface*](#) - receiving notifications regarding Mudra events: gestures, proportional and IMU orientation

c. IMudraAPI.aidl Interface

An interface used for controlling the Mudra connectivity / pairing status - scan for devices, connect\disconnect a device etc.

Member Functions:

void initMudra(IMudraDeviceStatusListener deviceCallback, IMudraDataListener dataCallback)

Initializes the SDK during binding, sets the Mudra Interface callback functions for asynchronous dataflow with all.

Parameters:

IMudraDeviceStatusListener deviceCallback – callback function for receiving device status notifications

IMudraDataListener dataCallback - callback function for receiving Mudra Events

Usage example in android:

```
import com.wearable.android.ble.interfaces.IAsyncDataListener;
import com.wearable.android.ble.interfaces.IAsyncDeviceStatusListener;
import com.wearable.android.ble.interfaces.MudraAPI;
```

```
private MudraAPI mMudraAPI = null;
```

@Override

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

//Binding Using Intents

```
    Intent intent = new Intent();
    intent.setAction(MudraAPI.class.getName());
    intent.setComponent(new ComponentName("com.wearable.android.ble",
"com.wearable.android.ble.service.BluetoothLeService"));
    getApplicationContext().bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
}
```

```
IMudraDataListener mMudraDataCB = new IMudraDataListener.Stub() {
    @Override
    public void onMudraDataReady (int dataType, float[] data) throws RemoteException {...}
}
```

```
IMudraDeviceStatusListener mMudraDeviceStatusCB = new IMudraDeviceStatusListener.Stub() {
    @Override
    public void onMudraStatusChanged(int statusType, String deviceAddress) throws RemoteException {...}
}
```

```
private ServiceConnection mConnection = new ServiceConnection() {
    // Called when the connection with the service is established using getApplicationContext()
    public void onServiceConnected(ComponentName className, IBinder service) {

        try {
```

```

        Log.i("INFO", "bind SUCCEEDED");
        // this gets an instance of the MudraAPI, which we can use to call on the service
        mIMudraAPI = MudraAPI.Stub.asInterface(service);
        mIMudraAPI.initMudra(mMudraDeviceStatusCB, mMudraDataCB);
    }
    catch (RemoteException ex)
    {
        Log.e("ERROR", ex.toString());
    }
}

// Called when the connection with the service disconnects unexpectedly
public void onServiceDisconnected(ComponentName className) {
    Log.e("ERROR", "Mudra Service has unexpectedly disconnected");
    mIMudraAPI = null;
}
};

```

void mudraStartScan()

Start scanning for Mudra devices. The scan duration is maximum 10 seconds and can be stopped using [mudraStopScan\(\)](#). The function results are readable with [onMudraStatusChange](#) callback. Run [initMudra](#) before calling this function.

Parameters:

No Parameters

Usage example in android:

```

public void StartScan (){

    try {
        mIMudraAPI.mudraStartScan();
    }
    catch (RemoteException ex)
    {
        Log.e("ERROR:", ex.toString());
    }
}

```

void mudraStopScan()

Stop scanning for Mudra devices after [mudraStartScan\(\)](#) was called. The function results are readable with [onMudraStatusChange\(\)](#) callback. Run [initMudra](#) before calling this function.

Parameters:

No Parameters

Usage example in android:

```
public void StopScan (){  
  
    try {  
        mIMudraAPI. mudraStopScan ();  
    }  
    catch (RemoteException ex)  
    {  
        Log.e("ERROR:", ex.toString());  
    }  
}
```

void connectMudraDevice(in String address)

Connect a Mudra device by its Bluetooth address. The function results are readable with [onMudraStatusChange\(\)](#) callback. Run [initMudra](#) before calling this function.

Parameters:

String Address - the relevant device Bluetooth address. for example: F3:DF:9b:43:26:14

Usage example in android:

```
public void ConnectMudra(deviceAddress){  
  
    try {  
        mIMudraAPI. connectMudraDevice(deviceAddress);  
    }  
    catch (RemoteException ex)  
    {  
        Log.e("ERROR:", ex.toString());  
    }  
}
```

void startRawSNCDDataTransmission()

Enable SNC signal transmission from Mudra device after it was connected. Run [initMudra](#) before calling this function.

Parameters:

No Parameters

Usage example in android:

```
public void startSNCDDataTransmission(){

    try {
        mlMudraAPI.startRawSNCDDataTransmission( );
    }
    catch (RemoteException ex)
    {
        Log.e("ERROR:", ex.toString());
    }
}
```

void stopRawSNCDDataTransmission()

Disable SNC signal transmission from Mudra device after it was connected. Run [initMudra](#) before calling this function.

Parameters:

No Parameters

Usage example in android:

```
public void stopSNCDDataTransmission(){

    try {
        mlMudraAPI.stopRawSNCDDataTransmission( );
    }
    catch (RemoteException ex)
    {
        Log.e("ERROR:", ex.toString());
    }
}
```

void disconnectMudraDevice()

Disconnect the current connected Mudra device. The function results are readable with [onMudraStatusChange\(\)](#) callback. Run [initMudra](#) before calling this function.

Parameters:

No parameters

Usage example in android:

```
public void disonnnectMudra(){

    try {
        mlMudraAPI.disconnectMudraDevice(deviceAddress);
    }
    catch (RemoteException ex)
    {
        Log.e("ERROR:", ex.toString());
    }
}
```

```
}  
}
```

void releaseMudra()

Releases the SDK and the async callbacks. Run [initMudra](#) before calling this function.

Parameters:

No parameters

Usage example in android:

```
public void releaseMudra(){  
  
    try {  
        mlMudraAPI.releaseMudra(deviceAddress);  
    }  
    catch (RemoteException ex)  
    {  
        Log.e("ERROR:", ex.toString());  
    }  
}
```

d. **IMudraDeviceStatusListener.aidl** Interface

An interface used for receiving notifications regarding Mudra status: scan results, device connected/disconnected, etc.

Member Functions:

`void onMudraStatusChanged (int statusType, in String deviceAddress)`

called by the mudra service whenever Mudra status changes

Parameters:

Int statusType – the notification type

PARAM VALUE	DEVICE STATUS
0	Device Scan Started
1	Device Scan Stopped
2	Device found in scan
3	Device connected
4	Device disconnected

String deviceAddress - the relevant device Bluetooth address. for example: F3:DF:9b:43:26:14

Usage example in android:

```
IMudraDeviceStatusListener mMudraDeviceStatusCB = new
IMudraDeviceStatusListener.Stub() {
    @Override
    public void onMudraStatusChanged(int statusType, String deviceAddress)
throws RemoteException {
    switch (statusType) {
        case0:
            Log.i ("INFO", "Device Scan Started");
            Break;
        case1:
            Log.i ("INFO", "Device Scan Stopped");
            Break;
        case2:
            Log.i ("INFO", "Device Found"+ deviceAddress);
            connectMudra(deviceAddress);
            Break;
        case3:
            Log.i ("INFO", "Device connected");
            startSNCDDataTransmission() {
            Break;
        case4:
            Log.i ("INFO", "Device disconnected");
            Break;
```

```
        }  
    }  
};
```


e. IMudraDataListener.aidl Interface

An Interface Used for receiving notifications regarding Mudra events: gestures, proportional and IMU orientation.

Member Functions:

void onMudraDataReady (int dataType, in float[] data)

called by the mudra service whenever an event is recognized

Parameters:

Int dataType – the notification type

Float[] data - float array with event related data

DATATYPE	EVENT		
0	Gesture Recognized	data[0]	Thumb gesture detected probability 0..1
		data[1]	Tap gesture detected probability 0..1
		data[2]	Index gesture detected probability 0..1
1	Proportional Recognized	data[0]	index is source of pressure probability 0..1
		data[1]	Middle finger is source of pressure probability 0..1
		data[2]	Applied force between 0..1
2	IMU event	data[0]	Acceleration X
		data[1]	Acceleration Y
		data[2]	Acceleration Z
		data[3]	Quaternion w
		data[4]	Quaternion X
		data[5]	Quaternion Y
		data[6]	Quaternion Z

Usage example in android:

```
IMudraDataListener mMudraDataCB = new IMudraDataListener.Stub() {
    @Override
    public void onMudraDataReady (int dataType, float[] data) throws
RemoteException {
        switch (dataType) {
            case 0:
                if ( (d[0] > d[1])&& (d[0]>d[2])&& (d[0]>0.9))
                    Log.i ("INFO", "gesture: Thumb");
                if ( (d[1] > d[0])&& (d[1]>d[2])&& (d[1]>0.9))
                    Log.i ("INFO", "gesture: Tap");
                if ( (d[2] > d[0])&& (d[2]>d[1])&& (d[2]>0.9))
                    Log.i ("INFO", "gesture: Index");
                Break;
            case 1:

```

```
        if ( d[0] > d[1])
            Log.i ("INFO", "Tap Proportional:" +data[2]);
        if ( d[1] > d[0])
            Log.i ("INFO", "Middle Tap Proportional:" +data[2]);
        Break;
    case 2:
        Log.i ("INFO", "IMU acc x: "+data[0]+
            " \nacc Y: "+data[1]+
            " \nacc Z: "+data[2]+
            " \nQ W: "+data[3]+
            " \nQ X: "+data[4]+
            " \nQ Y: "+data[5]+
            " \nQ Z: "+data[6]);
        Break;
    }
};
```

5. API - Unity

Will be documented in the next release.

6. API – Android Wear

Will be documented in the next release.

7. Use-Case Applications

We have developed a few demos and applications which demonstrate the current capabilities and use-cased for the Mudra device.

All applications can be downloaded from the Google Store.

a. Mudra AR App

Augmented Reality (AR) is a fast-growing field in gaming, marketing, work environment solutions etc. The Mudra AR Demo shows how the proportional can be used as a controller for certain AR applications.

Possible use-cases for the "proportional" may be:

- Drag and drop items – while the user applies any pressure
- Increase/Decrease – while the user increases/decreases pressure
- Zoon in/out – while the user prsses harder or softer

© Install app - <https://play.google.com/store/apps/details?id=com.mudrastrap.mudraar>

© Ready the beer coaster that you received with the sample shipment

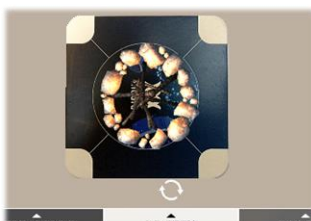
1. Running the Android application

- a. Open Mudra Capsule app and pair the device
- b. Go to the Signals tab and verify signal live stream
- c. Go to the Diagnostics tab, and verify that the proportional control display bar corresponds correctly with the pressure level you apply in "proportional" gesture

2. Running Mudra AR application

- a. Place the beer coaster on a well-lit table
- b. Open Mudra AR app and point the phone's camera on the beer coaster until you see an augmented fire placed on the coaster.
- c. Apply varying pressure between the index and thumb fingers and note how the height of the fire column changes in correspondence the pressure level
- d. You can change the augmented object from fire to beer glass tap "Take a test drive" on the bottom bar, and then tap the replace icon that is now visible.
- e. On the beer glass, the bubbles will climb faster the as pressure increases

click on the below images to view YouTube videos of this demo app.



b. Mudra VR App

Virtual Reality (VR) is gradually gaining consumers traction in gaming, social interactions and live performances. The Mudra VR Demo shows how the gestures and proportional can be used as a controller without the need to clench fingers around hand-held controllers, or raising your arms up in the air.

Possible use-cases may be:

- Drag and drop items – while the user applies any pressure
- Zoon in/out – while the user presses harder or softer
- Tap to select
- Thumb or index to navigate through a menu

- ☉ Install app - <https://play.google.com/store/apps/details?id=com.mudraptrap.wld>
- ☉ Ready a simple VR headset, such as Google cardboard and alike

1. Running the Android application
 - a. Open Mudra Capsule app and pair the device
 - b. Go to the Signals tab and verify signal live stream
 - c. Go to the Diagnostics tab, and verify that the proportional control display bar corresponds correctly with the pressure level you apply in "proportional" gesture
2. Running Mudra VR application
 - a. Open Mudra VR app
 - b. Place the phone into the headset and wear the headset
 - c. use "proportional" to move around – you will advance in the direction that you are looking at a speed proportional to the pressure you apply
 - d. Look for boxes laid around. When the crosshair lays on a box it will turn into circle. "tap" and the box will explode
 - e. Find the keyhole on the control panel. When the crosshair lays on it, it turns into circle. Tap to open the control panel and interact with the different controls and input methods
 - f. Look for a pile of boxes. Tap to select a box, and then move it around and rotate it by moving your hand. Tap again to drop the box in its new place. "drag and drop"

click on the below images to view YouTube videos of this demo app.



c. Mudra Smartwatch App

Smartwatches are gradually gaining their place among consumers as an auxiliary device next to the smartphone, and nowadays some of them provide independent connectivity, which supports many sports and on-the-go or activities. The Mudra Smartwatch Demo shows how the gestures and proportional control a smartwatch using same-hand subtle finger movements.

Possible use-cases may be:

- Swipe or scroll in two directions using thumb and index
- Select using tap
- Increase/decrease volume using proportional

- ⦿ Install Mudra watch app -
- ⦿ Install Mudra watch face –
- ⦿ Install Mudra battery monitor -

1. Running the Android application
 - a. Open Mudra Capsule app and pair the device
 - b. Go to the Signals tab and verify signal live stream
 - c. Go to the Diagnostics tab, and verify that the proportional control display bar corresponds correctly with the pressure level you apply in "proportional" gesture
2. Verify that your smartwatch is connected to your phone
3. Replace your Watch Face with the Mudra watch face
4. Push the watch button to go to apps and select the Mudra watch app. Tap the toggle button to turn on gesture control. Now when you make a gesture the gesture number will appear on the screen
5. Running the watch demo:
 - a. When the watch shows the Mudra watch face, do the tap gesture to enter the mudra music player.
 - b. Use the thumb and index finger to navigate between the media player screens and tap to select the property in the screen.
 - c. Go to the volume screen and use the pressure between your fingers to control the Music amplitude.

click on the below images to view YouTube videos of this demo app.



8. FAQ and Known Issues

Q: I cannot connect to the Mudra band through the Mudra app.

A: Charge the band using and make sure a green led is blinking after pressing restart. Make sure the Bluetooth is enabled on the phone.

Q: I can see a noticeable lag in the signals when viewing the “Signals” tab.

A: Make sure wifi is disabled or at least not performing download and updates. In addition, make sure your phone is not multitasking too heavily, close other unnecessary apps and restart the application.

Q: Gesture recognition is constantly firing! There are a lot of false recognitions.

A: Make sure the Mudra band is in contact with your skin. Also, relax your arm and wrist, Mudra works well on gentle gestures. Finally, in the “Signals” tab, check out how your signals look when your hand is relaxed and when you are performing a gesture.

Q: I'm getting a lot of errors in gesture recognition after calibration.

A: The current calibration algorithm is not robust enough and will not fit all users. Please help us by performing calibration multiple times so we can use your data to make a better model! Also, make sure your hand is relaxed during gesture calibration and usage and that you are following the gesture guidelines in the calibration videos.

Q: My Mudra app crashed.

A: Please contact support (<mailto:support@wearabledevices.co.il>). We will work with you to resolve your issue.

Q: I would like to conduct research in signal processing and machine learning (deep learning etc.). How can I use Mudra to conduct biopotential R&D?

A: We provide a turnkey solution for R&D. This includes all SW for deploying signal processing and machine learning algorithms on an Android phone. We also provide Tensorflow support for real-time deep learning model inference on Android. Finally, simulations for analyzing and deploying Tensorflow models can also be provided. Contact us for more info <mailto:support@wearabledevices.co.il>.

9. Contact Details

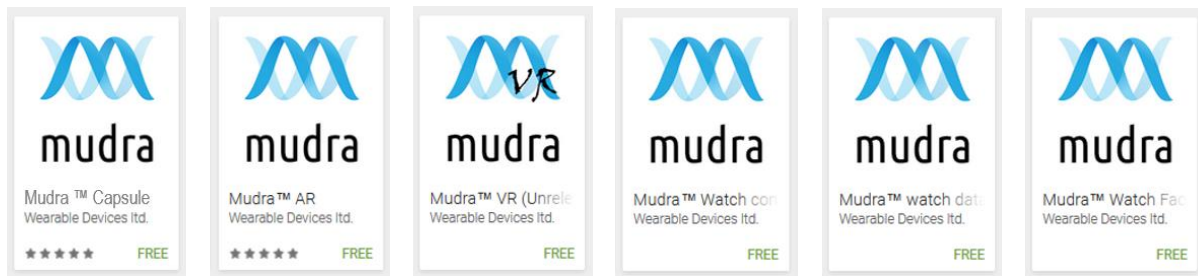
For any other issues or questions please contact <mailto:support@wearabledevices.co.il>

Sources

For any other issues or questions please contact <mailto:support@wearabledevices.co.il>

Applications

All Wearable Devices LTD android apps can be downloaded using the following [link to the App store](#).



Documentation

Most updated Wearable Devices LTD related API and SDK documentation can be downloaded using the following [link to a Google Drive Folder](#).

