



moticon OpenGo Sensor Insoles Wireless Pressure Force And Motion Sensing User Guide

[Home](#) » [moticon](#) » moticon OpenGo Sensor Insoles Wireless Pressure Force And Motion Sensing User Guide



moticon OpenGo Sensor Insoles Wireless Pressure Force And Motion Sensing



Contents

- 1 Disclaimer
- 2 Developing with OpenGo SDKs
- 3 Overview
 - 3.1 System Architecture
 - 3.2 Available SDKs
 - 3.3 Sensor Data
 - 3.4 Communication Messages
- 4 OpenGo Endpoint SDK
 - 4.1 System Overview
 - 4.2 Required Components
- 5 OpenGo Insole SDK
 - 5.1 System Overview
 - 5.2 Required Components
- 6 OpenGo Mobile SDK
 - 6.1 System Overview
 - 6.2 Required Components
- 7 Feature Overview
- 8 Programming Challenges
- 9 A Decision Guide
- 10 Legal Note and Disclaimer
- 11 Changelog
- 12 Customer Support
- 13 Documents / Resources
 - 13.1 References
- 14 Related Posts

Disclaimer

The Software Development Kits (SDKs) provided by Moticon are designed for maximum access and utility of the sensor insole hardware. However, prior to start working with the SDKs, please note that there are technical limitations as well. It is the obligation of the user/customer to clarify these and other technical limitations before entering corresponding developments and projects:

- Platforms: The SDKs may not work on specific desktop/mobile platforms. Also, please note the minimum system requirements provided on moticon.com/opengo/faqs.
- Transmission delays: Data transmission delays will occur wherever data is buffered or (re-) transmitted. Moticon cannot determine how large the resulting overall transmission delay is in a given environment (in mean or worst-case sense), and whether or not this is within acceptable bounds for the intended application.
- Radio connection: The radio connection is only provided within a range inherent to the BLE technology, and largely depends on the particular environment. The vicinity of both sending and receiving units, the radio propagation path between them, and interfering radio equipment has immediate impact on the radio connection quality.
- Connection stability: The flow of control information and measurement data is affected by layers which cannot all be strictly controlled. For example, system functions of mobile phones may cause radio connections to terminate after defined or undefined amounts of time, affecting long-term data transmission. Such problems can be specific to the mobile phones and network equipment in use.

Developing with OpenGo SDKs

The Moticon sensor insole ecosystem is open for custom application development utilizing the sensor insole hardware.

The sensor insoles use Bluetooth BLE radio technology for coupling with mobile phones and tablets, yet further software components are required to control sensor insoles and make use of the sensor data. Moticon provides the required components as described in this guide.

For mobile applications, currently only Android is supported.

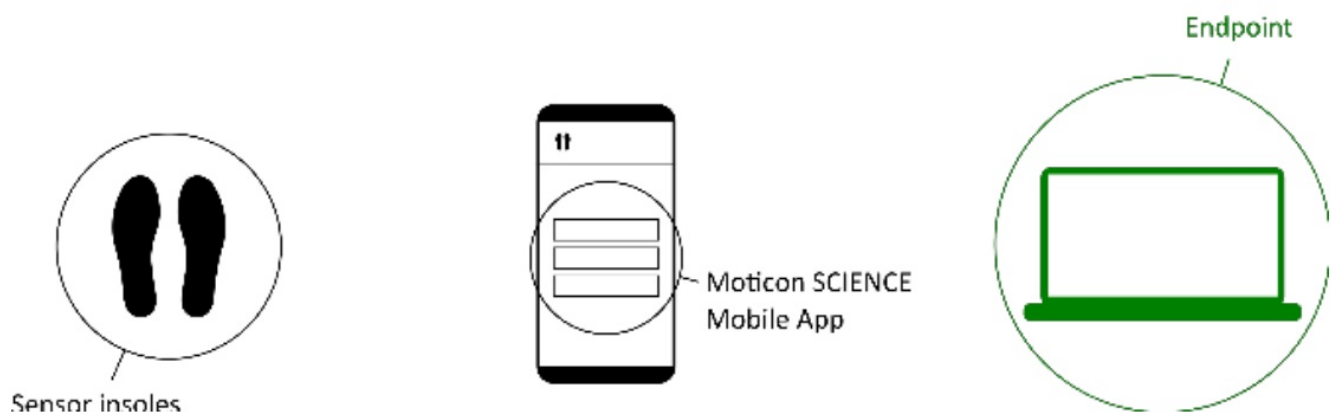
Note: You may start the development of data processing algorithms simply by working with plain text sensor insole data, which can be exported using the Moticon OpenGo Software. The Moticon OpenGo Software even provides a real-time UDP export of data captured live by sensor insoles, e.g. for prototyping feedback applications. Beyond such data processing, any software development that involves active control of sensor insoles requires the components described in this guide.

Overview

System Architecture

A Moticon sensor insole application typically comprises the following components:

Figure 1 — OpenGo system components



As a programmer, you will typically work with

- Moticon sensor insoles
- the Moticon insole3 service, an Android library
- a mobile app (your own app or the Moticon OpenGo Mobile App)
- a data endpoint (e.g. PC software)

Depending on the pursued application and the corresponding SDK (see Sec. 3.2), only some of the above components may be part of your solution.

Available SDKs

The OpenGo SDKs allow you to develop custom solutions in three different scenarios:

1. Endpoint SDK (Sec. 4): Develop your own endpoint solution, which could e.g. be a lab measurement system

where a local computer processes live sensor data. Your part is the software running on the endpoint system (computer), while the sensor insoles are controlled using the standard Moticon OpenGo Mobile App.

2. Insole SDK (Sec. 5): Take over control of the sensor insoles and receive sensor data in real time, even without need for using a mobile phone. This can be integrated into various platforms supporting BLE, including mobile, Linux, and single-board computers.
3. Mobile SDK (Sec. 6): Develop your own mobile app, which could e.g. be an app for starting and stopping measurements, and transforming data from Moticon sensor insoles into audio feedback. The insole3 service takes over the communication with the sensor insoles via BLE, and thereby speeds up mobile app development.

All SDKs can be used independent of each other. If using the Moticon OpenGo Mobile App is still an option, the Endpoint SDK will allow for fastest results and lowest programming efforts.

The Insole SDK applies to cases where no mobile phone shall be used at all. It only supports live data capture, no on-insole recording.

The Mobile SDK can be considered as abstraction layer, which integrates into Android apps for simplifying the sensor insole control. It supports both live data capture and recording.

Sensor Data

Moticon sensor insoles can measure and transmit various kinds of sensor data:

- 16 pressure sensors (in 1/4 N/cm²)
- 3 acceleration axes (in g)
- 3 angular rate axes (in degree/s)
- 1 total force value (in N)
- 2 center of pressure (COP) axes (in percent of insole length/width)

In order to save memory space and battery, the user may select only a subset of the above data channels by using the Moticon OpenGo Mobile App or the SDKs.

The selected data channels are communicated in corresponding messages of the communication protocol. The data messages then contain the corresponding data channels, along with the relative time in milliseconds.

Communication Messages

For exchanging messages, all system components use Google's Protocol Buffers for encoding (developers.google.com/protocol-buffers/).

By compiling the files `common.proto`, `service.proto`, and `insole.proto` (requires respective SDKs) for your source code using a protocol buffer compiler, you will be able to interpret the binary data messages which are exchanged according to the communication protocol:

- For a live data endpoint (Endpoint SDK), the messages received from the Moticon OpenGo Mobile App are proto messages.
- For direct interaction with sensor insoles (Insole SDK), the messages exchanged via BLE are proto messages.

- For a mobile app (Mobile SDK), the messages exchanged with the insole3 service are also proto messages.

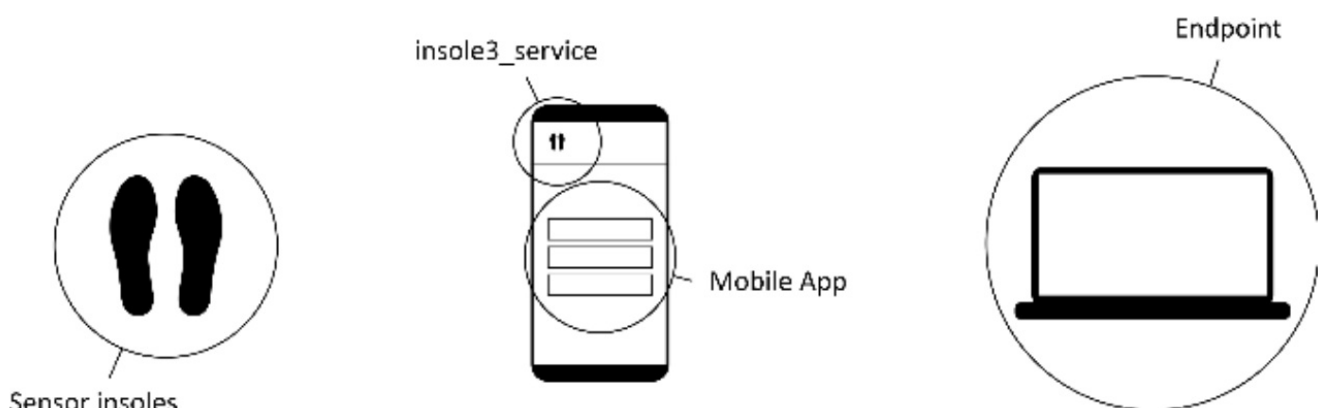
A data endpoint will receive sequences of proto messages. The messages are separated in the TCP/BLE data flow by length-prefix framing, where each message is prefixed with the message length, encoded as 2-byte big-endian integer value.

OpenGo Endpoint SDK

System Overview

The data transmitted by Moticon sensor insoles is forwarded by the Moticon OpenGo Mobile App, and is finally received by a so called endpoint. The endpoint is typically a desktop software or cloud server.

Figure 2 — Custom endpoint implementation



A simple endpoint implementation may be a Python script running a TCP server. The server's IP address and port number can be set in the Moticon OpenGo Mobile App, in order to let the Moticon OpenGo Mobile App forward the data received from the sensor insoles to the endpoint server.

In this scenario, the workflow is to

1. Use the Moticon OpenGo Mobile App to pair sensor insoles.
2. Configure the connection to your endpoint (IP address and port) in the Moticon OpenGo Mobile App.
3. Start a live capture measurement using the Moticon OpenGo Mobile App.
4. Process the received data in your endpoint software.

This requires you to understand the communication protocol which your endpoint software has to implement in order to interact with the Moticon OpenGo Mobile App.

Required Components

For receiving live data from Moticon sensor insoles in a custom endpoint, you will need:

- Moticon sensor insoles
- Moticon OpenGo Mobile App
- The protobuf files `common.proto` and `service.proto`
- A data endpoint programmed by you

Compiling the files `common.proto` and `service.proto` provides stubs for interpreting the binary data messages forwarded by the Moticon OpenGo Mobile App in your endpoint implementation.

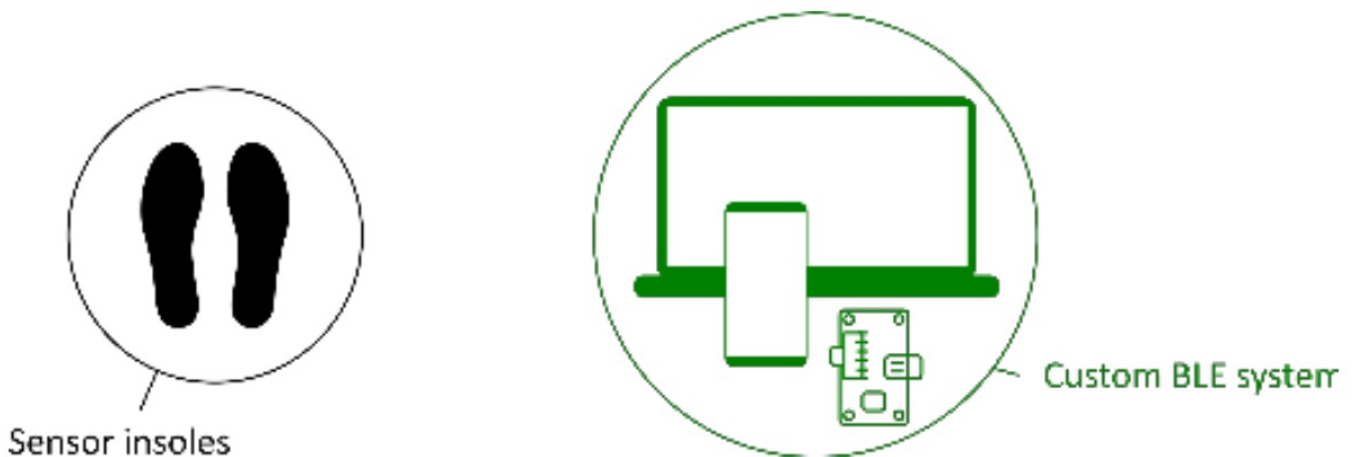
The SDK contains a working Python endpoint sample implementation.

OpenGo Insole SDK

System Overview

The sensor insole control and transmission of live capture data is facilitated by an interface description based on `.proto` files.

Figure 3 — Custom BLE system implementation



This scenario requires you to:

1. Develop your own software able to send and receive via BLE.
2. Implement a communication protocol based on the message types defined in the provided `.proto` files.

The SDK is intended for advanced programmers familiar with BLE system development.

Required Components

For direct communication with the sensor insoles via BLE, you will need:

- Moticon sensor insoles
- A BLE dongle, integrated BLE electronics, or a BLE-enabled single-board computer, along with a BLE software stack
- The protobuf files `common.proto` and `insole.proto`
- A software programmed by you, able to pair devices and send/receive data via the systemspecific BLE software stack

The Insole SDK is the most low-level OpenGo SDK. Please note that, while even mobile apps can be created using the Insole SDK, all basic functionalities have to be custom programmed, e.g. keeping a live data stream running as foreground service when the mobile phone screen is locked. The Insole SDK is a versatile solution for

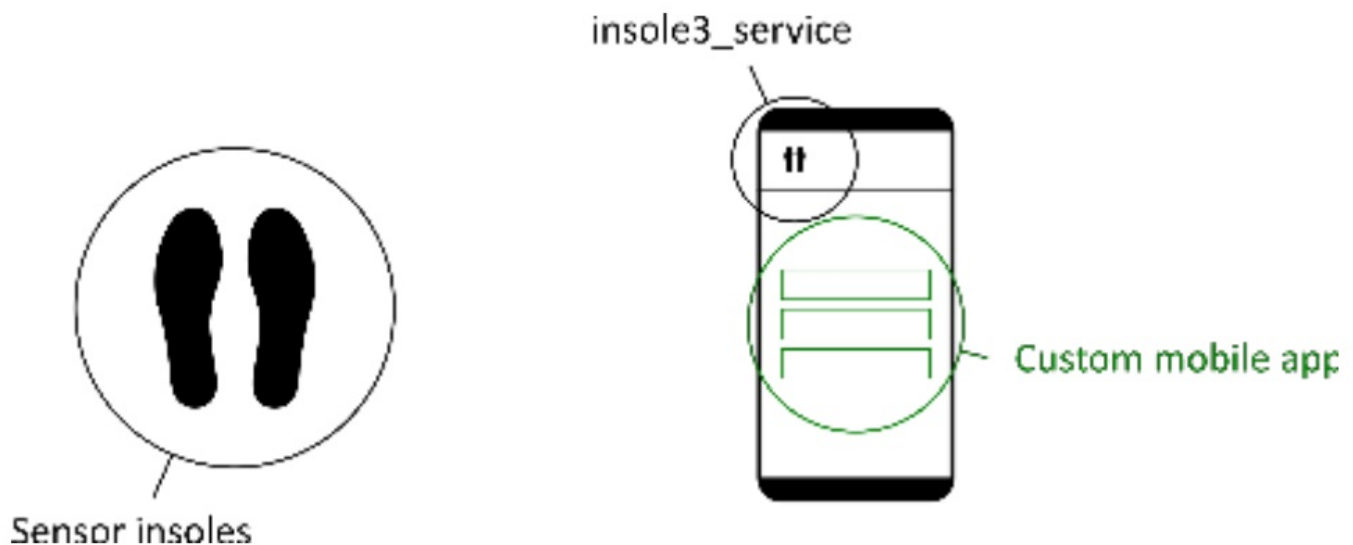
applications requiring a maximum integration level of OpenGo, such as applications running on single-board computers.

OpenGo Mobile SDK

System Overview

The sensor insole control and data transmission is facilitated by an Android library provided by Moticon, called insole3 service. The insole3 service is used by the mobile app, and will start a foreground service for continuing an ongoing data transfer if the app is minimized.

Figure 4 — Custom mobile app implementation



This scenario requires you to:

1. Develop your own Android application and deploy it on your mobile device.
2. Understand how to include the insole3 service in your app.
3. Use all required functionality of the insole3 service, such as searching for sensor insoles, handling data measurement services, etc.
4. Process and/or forward the sensor data streamed by the insole3 service.
5. React to events reported by the insole3 service, e.g. sensor insole disconnect events

The SDK is intended for advanced programmers familiar with mobile app development.

Required Components

For controlling Moticon sensor insoles from your own mobile app, you will need:

- Moticon sensor insoles
- The Moticon insole3 service, provided as file moticon insole3 service-xx xx xx-release.aar
- The protobuf files common.proto and service.proto
- An Android app programmed by you

The insole3 service is used by including the library in Gradle, and adding services to the Manifest. In your app code, Android permissions and checks shall be carried out. After binding (and later unbinding) the service, proto messages can be sent and received.

The SDK contains code examples and flowcharts for explaining all these steps in detail.

Feature Overview

The following table is an overview of the features provided by each of the different OpenGo SDKs:

Table 1 — Feature overview.

	Endpoint SDK	Insole SDK	Mobile SDK
Feature			
Runs on embedded platforms Runs on Android	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Start/stop live data Receive live data Forward live data to OpenGo Software Receive live data on PC (Windows, OSX, Linux)	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Start/stop recording Forward recording data to OpenGo Software			<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Sensor insole handling (scan/connect/disconnect) Manual zeroing		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Sensor insole calibration (as in OpenGo App)			
Report parameters (gait, jump etc. as in OpenGo Software)			

Programming Challenges

Before deciding for software development with an OpenGo SDK, please make sure that you can cover the required resources and skills.

So please carefully ask yourself:

- Do you have/are a senior software developer?

- Is your team experienced in all techniques required for the different SDKs, and for testing, deploying, and trouble-shooting solutions on the target platform?
- Are you capable of developing reliable communication systems based on sample message flows?
- If developing for Android, are you familiar with Android Broadcast Intents?

Moticon does not provide programming support. Due to the large variety of platforms and languages, Moticon also does not provide ready-to-use code (except for an endpoint Python example).

The SDKs come with enough documentation as far as the use of Moticon components is concerned. Each of the SDKs has already been successfully used by a large number of customers.

However, there will be no general documentation on topics like:

- Protocol buffers (protobuf)
- Programming languages (Python, Java/Kotlin, Dart, or any other)
- Integration into specific OS'es
- Third party SDKs, frameworks, and components (mobile app development, BLE sticks and drivers, etc.)
- Device-specific BLE handling

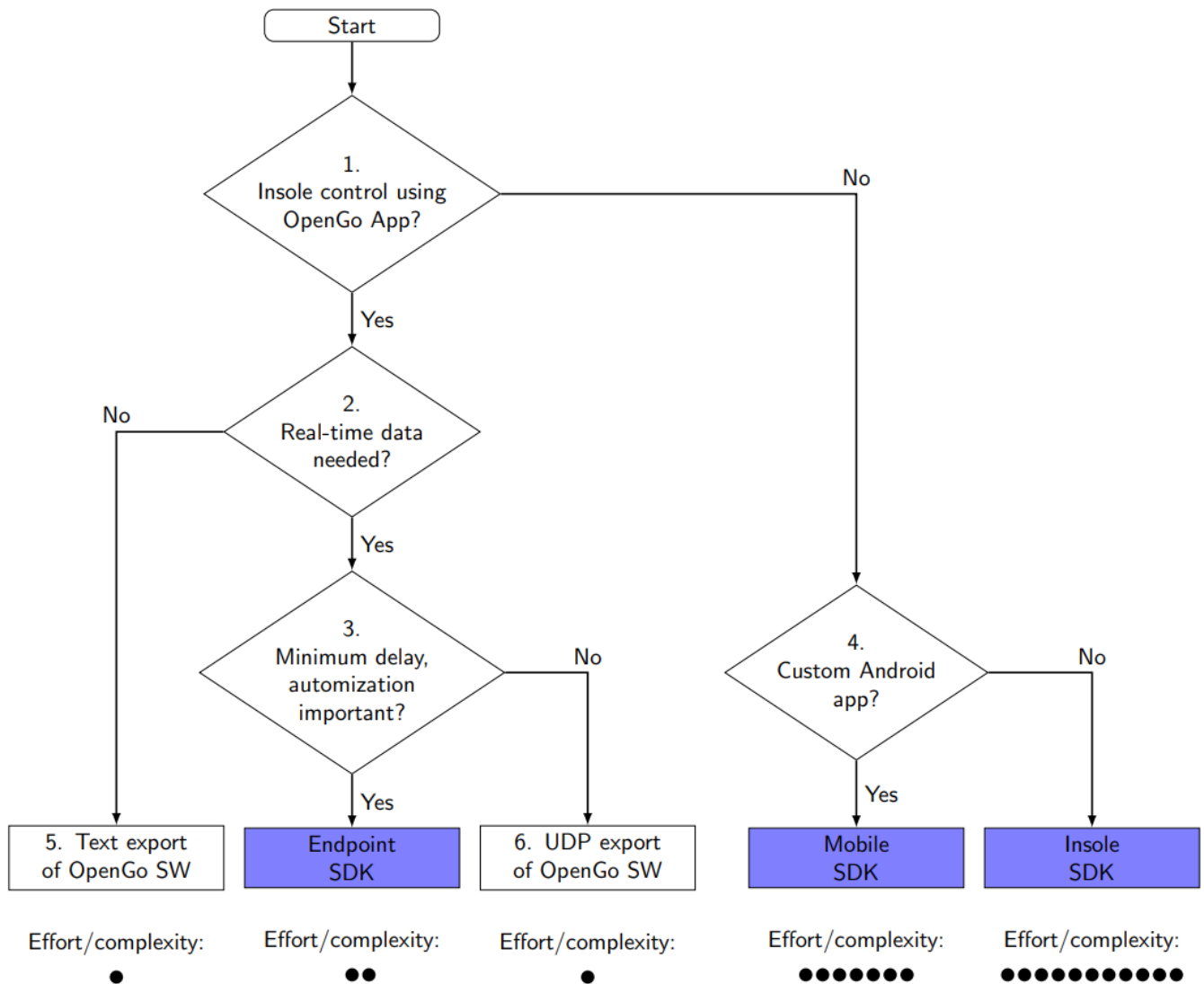
Generally speaking, the Insole SDK is the most ambitious solution, followed by the Mobile SDK.

Moticon cannot guarantee that the intended system solution, behavior, and performance can be accomplished by the SDK in question. Also, please note that certain functionalities of the standard Moticon OpenGo products are not available in the SDKs, e.g. calibration to body weight.

If in doubt, please discuss your intended development project with the Moticon support team.

A Decision Guide

The following decision tree shall help you in finding the right OpenGo solution for your project. Note, however, that additional aspects may impact your decision. For example, the sensor insole pressure sensor calibration is currently only available by using the Moticon OpenGo Mobile App.



Comments (numbers as in above figure):

1. The primary question is whether it is acceptable to use the standard Moticon OpenGo Mobile App for controlling the sensor insoles (connect, start/stop, data channel selection, etc.). If possible, this greatly simplifies the overall project. This of course requires the manual operation of the mobile device running the Moticon OpenGo Mobile App, which may not be possible in automated setups.
2. Another aspect to consider is whether the sensor insole data needs to be available in realtime, i.e. for sample-by-sample processing parallel to the measured motion.
3. The next question is whether the delay (and possibly also packet loss) caused by an additional UDP transmission is acceptable, and whether there is a need for a custom solution eliminating the Moticon OpenGo Software. The UDP export configuration, as well as the receiving script, also have to be adapted each time the sensor setup is changed in the Moticon OpenGo Mobile App. In contrast, the Endpoint SDK receives the data in a structured, responsive way.
4. If the Moticon OpenGo Mobile App cannot be used, the question is whether a custom Android app shall be developed, or instead a different (custom) BLE system is used.
5. For collecting offline data – no matter if the sensor insoles have been operated in live capture or recording – the simplest solution is to collect data using the standard OpenGo system, and export the data from the Moticon OpenGo Software as text file.
6. The Record section of the Moticon OpenGo Software can be configured to forward any received data message

via UDP. Please refer to the Moticon OpenGo Software documentation for further details

Legal Note and Disclaimer

Copyright (c) 2022, Moticon ReGo AG All rights reserved.

Any redistribution in source or binary forms, with or without modification, is not permitted.

THIS SOFTWARE IS PROVIDED BY MOTICON REGO AG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL MOTICON REGO AG OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Changelog

Version	Date	Changes
3.3	18.08.2022	Add decision guide section.
3.2	19.11.2021	Add feature overview section.
3.1	21.10.2021	Add programming challenges section.
3.0	01.09.2021	Change product name from SCIENCE to OpenGo.
2.0	23.03.2020	Add disclaimer. Describe different programming SDKs.
0.1	04.02.2019	Draft version.


Customer Support

Contact

Moticon ReGo AG
Address: Machtlfinger Str. 21, 81379 Munich, Germany
Email: support@moticon.com
Phone: +49 89 2000 301 50



Documents / Resources

 <small>Application Note 002 OpenGo Programmer's Guide</small>	moticon OpenGo Sensor Insoles Wireless Pressure Force And Motion Sensing [pdf] User Guide OpenGo Sensor Insoles Wireless Pressure Force And Motion Sensing, OpenGo, Sensor Insole s Wireless Pressure Force And Motion Sensing, Wireless Pressure Force And Motion Sensing, Pressure Force And Motion Sensing, Force And Motion Sensing, Motion Sensing
--	--

References

- [🔗 OpenGo frequently asked technical questions](#)