



## **Application Note 002**

OpenGo Programmer's Guide

## Contact

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

## Legal Note and Disclaimer

Copyright (c) 2025, 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
3.4	24.06.2025	Remove Mobile SDK.
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.

## Contents

<b>1 Disclaimer</b>	<b>4</b>
<b>2 Developing with OpenGo SDKs</b>	<b>4</b>
<b>3 Overview</b>	<b>5</b>
3.1 System Architecture . . . . .	5
3.2 Available SDKs . . . . .	5
3.3 Sensor Data . . . . .	6
3.4 Communication Messages . . . . .	6
<b>4 OpenGo Endpoint SDK</b>	<b>7</b>
4.1 System Overview . . . . .	7
4.2 Required Components . . . . .	7
<b>5 OpenGo Insole SDK</b>	<b>8</b>
5.1 System Overview . . . . .	8
5.2 Required Components . . . . .	8
<b>6 Feature Overview</b>	<b>9</b>
<b>7 Programming Challenges</b>	<b>10</b>
<b>A Decision Guide</b>	<b>11</b>

## 1 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 <https://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.

## 2 Developing with OpenGo SDKs

The Moticon OpenGo Sensor Insole ecosystem is designed to be open for custom application development, leveraging the sensor insole hardware.

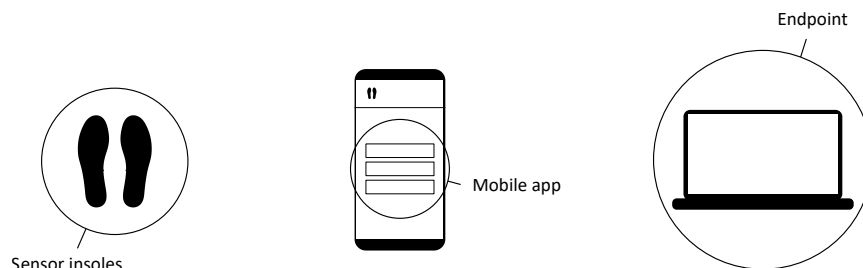
The sensor insoles utilize Bluetooth BLE radio technology for connection with controlling devices. However, additional software components are necessary to control the sensor insoles and effectively use the sensor data. Moticon provides these essential components, as detailed in this guide.

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

## 3 Overview

### 3.1 System Architecture

A typical Moticon OpenGo Sensor Insole application generally comprises the following components:



**Figure 1** — OpenGo system components

As a programmer, your work will typically involve:

- Moticon OpenGo Sensor Insoles
- A mobile application (either your custom app or the Moticon OpenGo App)
- A data endpoint (e.g., PC software)

Depending on the specific application and the corresponding SDK chosen (refer to Sec. 3.2), only a subset of the above components may be required for your solution.

### 3.2 Available SDKs

The OpenGo SDKs empower you to develop custom solutions across two distinct scenarios:

1. **Endpoint SDK (Sec. 4):** Develop your own endpoint solution. This could be, for instance, a lab measurement system where a local computer processes live sensor data. Your primary responsibility is the software running on the endpoint system (computer), while the sensor insoles are controlled using the standard Moticon OpenGo App.
2. **Insole SDK (Sec. 5):** Assume direct control of the sensor insoles and receive real-time sensor data, eliminating the need for a mobile phone. This SDK can be integrated into various BLE-supporting platforms, including mobile devices, Linux systems, and single-board computers.

These SDKs can be used independently. If leveraging the Moticon OpenGo App remains a viable option for your project, the Endpoint SDK will generally lead to the fastest results with the least programming effort.

The Insole SDK is applicable to cases where a custom mobile app (Android/iOS) will control the sensor insoles, as well as to systems where a mobile phone is not part of the solution at all.

### 3.3 Sensor Data

Moticon OpenGo Sensor Insoles are capable of measuring and transmitting various types of sensor data:

- 16 pressure sensors (in  $1/4 \text{ N/cm}^2$ )
- 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)

To optimize memory usage and battery life, users can select a subset of the above data channels using either the Moticon OpenGo App or the SDKs.

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

### 3.4 Communication Messages

For message exchange, all system components utilize Google's Protocol Buffers for encoding (<https://developers.google.com/protocol-buffers/>).

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

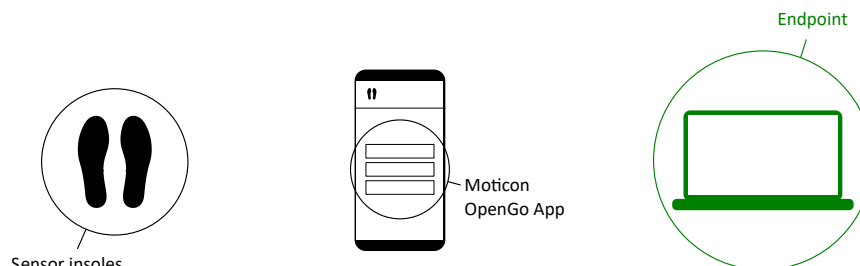
- For a live data endpoint (Endpoint SDK), the messages received from the Moticon OpenGo App are Protocol Buffer messages.
- For direct interaction with sensor insoles (Insole SDK), the messages exchanged via BLE are Protocol Buffer messages.

A data endpoint will receive sequences of Protocol Buffer messages. These messages are delimited within the TCP/BLE data flow by length-prefix framing, where each message is prefixed with its length, encoded as a 2-byte big-endian integer value.

## 4 OpenGo Endpoint SDK

### 4.1 System Overview

The data transmitted by Moticon OpenGo Sensor Insoles is forwarded by the Moticon OpenGo App and ultimately received by what is referred to as an *endpoint*. This endpoint is typically a desktop application or a cloud server.



**Figure 2** — Custom endpoint implementation

A straightforward endpoint implementation could be a Python script running a TCP server. The server's IP address and port number can be configured within the Moticon OpenGo App allowing the Moticon OpenGo App to forward the data received from the sensor insoles to your endpoint server.

In this scenario, the typical workflow involves:

1. Using the Moticon OpenGo App to pair sensor insoles.
2. Configuring the connection to your endpoint (IP address and port) within the Moticon OpenGo App.
3. Initiating a live capture measurement using the Moticon OpenGo App.
4. Processing the received data within your endpoint software.

This process necessitates an understanding of the communication protocol that your endpoint software must implement to interact with the Moticon OpenGo App.

### 4.2 Required Components

To receive live data from Moticon OpenGo Sensor Insoles in a custom endpoint, you will need:

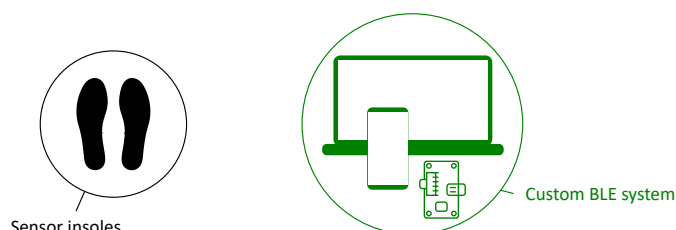
- Moticon OpenGo Sensor Insoles
- The Moticon OpenGo App
- The Protocol Buffer definition files `common.proto` and `service.proto`
- A data endpoint application developed by you

Compiling the files `common.proto` and `service.proto` provides the necessary stubs for interpreting the binary data messages forwarded by the Moticon OpenGo App within your endpoint implementation. The SDK includes a functional Python endpoint sample implementation to assist your development.

## 5 OpenGo Insole SDK

### 5.1 System Overview

Control of the sensor insoles and transmission of live capture data are 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 capable of scanning for BLE devices and sending/receiving BLE data packets.
2. Implement a communication protocol based on the message types defined in the provided .proto files.

This SDK is intended for experienced programmers familiar with BLE system development.

### 5.2 Required Components

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

- Moticon OpenGo Sensor Insoles
- A BLE dongle, integrated BLE electronics, or a BLE-enabled single-board computer, along with a compatible BLE software stack
- The Protocol Buffer definition files `common.proto` and `insole.proto`
- A software application developed by you, capable of scanning for BLE devices and sending/receiving data via the system-specific 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 must be custom programmed. This includes, for example, maintaining a live data stream as a foreground service when the mobile phone screen is locked.

The Insole SDK offers a versatile solution for applications requiring the maximum level of OpenGo integration, such as applications running on single-board computers.

## 6 Feature Overview

The following table provides an overview of the features supported by each of the different OpenGo SDKs:

Feature	Endpoint SDK	Insole SDK
Runs on embedded platforms	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Suitable for app development	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Start/stop live data	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Receive live data	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Forward live data to OpenGo Software	<input type="checkbox"/>	<input type="checkbox"/>
Start/stop recording	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Transfer and decode recording data	<input type="checkbox"/>	<input type="checkbox"/> <sup>1</sup>
Sensor insole handling (scan/connect/disconnect)	<input type="checkbox"/>	<input type="checkbox"/>
Manual zeroing	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Sensor insole calibration <sup>2</sup>	<input type="checkbox"/>	<input type="checkbox"/>
Report parameters (gait, jump etc. as in OpenGo Software)	<input type="checkbox"/>	<input type="checkbox"/>

**Table 1** — Feature overview.

<sup>1</sup>Available upon request

<sup>2</sup>The Moticon OpenGo App can be used for calibration prior to SDK-based data acquisition

## 7 Programming Challenges

Before committing to software development with an OpenGo SDK, please ensure that you have access to the necessary resources and skills.

Therefore, we urge you to carefully consider the following questions:

- Do you have a senior software developer on your team, or are you one yourself?
- Is your team proficient in all techniques required for the different SDKs, and for testing, deploying, and troubleshooting solutions on the target platform?
- Are you capable of developing robust communication systems based on sample message flows?
- If developing for a mobile platform, are you familiar with platform-specific permission handling and BLE communication?

Moticon does not provide direct programming support. Due to the wide variety of platforms and languages, Moticon also does not offer ready-to-use code, beyond basic Python examples.

The SDKs are accompanied by sufficient documentation concerning the use of Moticon components. Each of the SDKs has already been successfully utilized by a large number of our customers.

However, please be aware that general documentation on topics such as the following will not be provided:

- Protocol buffers (protobuf)
- Specific programming languages (Python, Java/Kotlin, Dart, or any other)
- Integration into specific operating systems
- Third-party SDKs, frameworks, and components (e.g., mobile app development, BLE dongles and drivers)
- Device-specific BLE handling

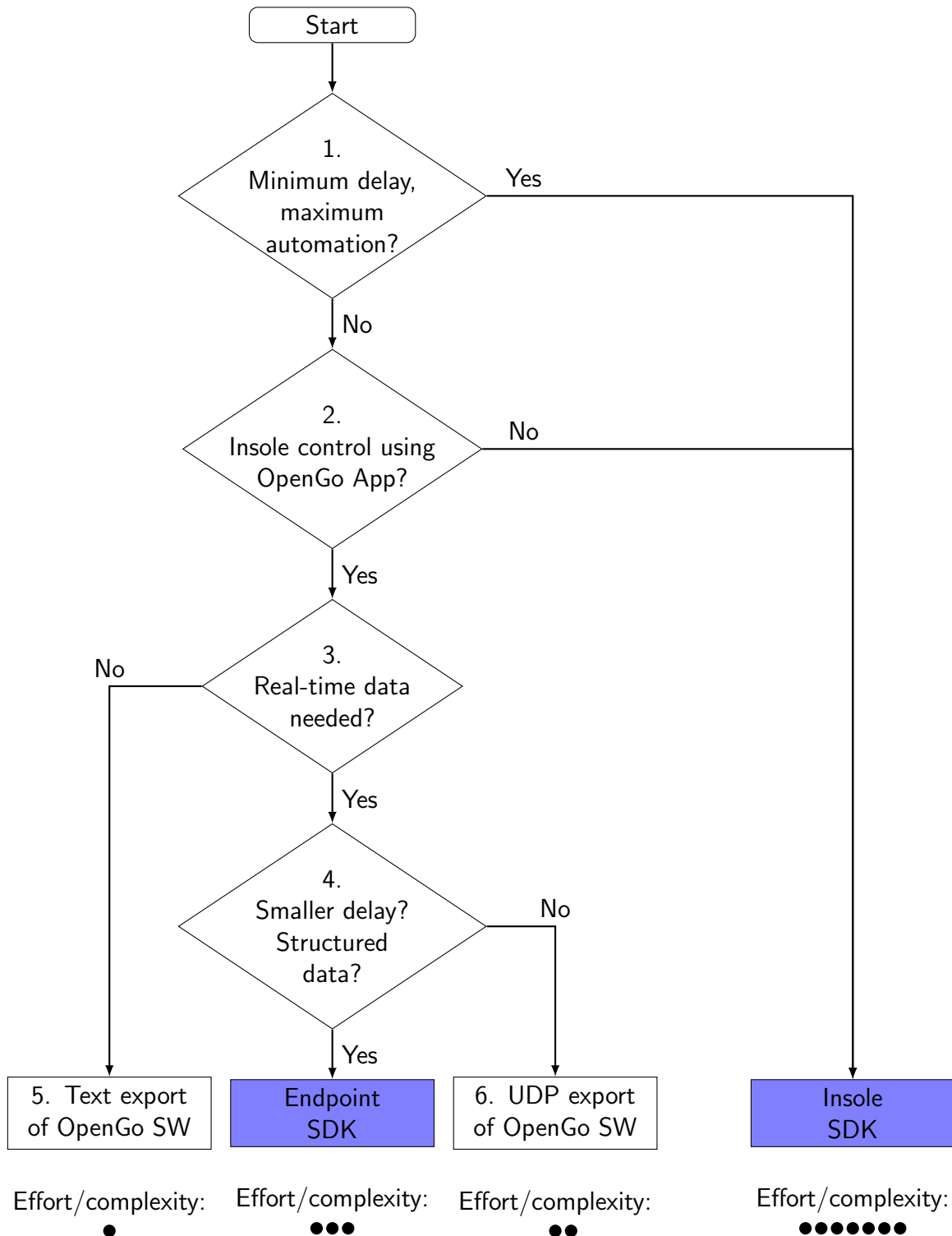
Generally speaking, the Insole SDK represents the most ambitious development solution.

Moticon cannot guarantee that your intended system solution, behavior, and performance can be fully achieved using the SDK in question. Additionally, please note that certain functionalities of the standard Moticon OpenGo products, such as calibration to body weight, are not available in the SDKs.

If you have any doubts, please discuss your intended development project with the Moticon support team.

## A Decision Guide

The following decision tree is designed to assist you in selecting the most suitable OpenGo solution for your project. Please note, however, that additional factors may influence your decision. For example, the sensor insole pressure sensor calibration is currently only available when using the Moticon OpenGo App.



**Comments (numbers as in above figure):**

1. Determine whether your solution has critical delay requirements or aims for deep integration of sensor insoles (e.g., automated start/stop commands).
2. A key consideration is whether it is acceptable to use the standard Moticon OpenGo App for controlling the sensor insoles (connecting, starting/stopping measurements, selecting data channels, etc.). If feasible, this significantly simplifies the overall project. Naturally, this requires manual operation of the mobile device running the Moticon OpenGo App, which may not be suitable for automated setups.
3. Another aspect to consider is whether the sensor insole data needs to be available in real-time, i.e., for sample-by-sample processing concurrently with the measured motion.
4. The next question is whether the delay (and potential packet loss) introduced by an additional UDP transmission is acceptable, and whether there is a need for a custom solution that bypasses the Moticon OpenGo Software. The UDP export configuration, as well as the receiving script, must also be adjusted each time the sensor setup is modified in the Moticon OpenGo App. In contrast, the Endpoint SDK receives data in a structured and responsive manner.
5. For collecting offline data, regardless of whether the sensor insoles were operated in live capture or recording mode, the simplest solution is to collect data using the standard OpenGo system and export it from the Moticon OpenGo Software as a 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 more detailed information.