Application Note 009

OpenGo Propagation Delay

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

Changelog

Version	Date	Changes
1.0	27.09.2022	Initial version

Contents

1 General Notes

This application note describes the signal propagation delay of the OpenGo system. The given delay values characterize the observed latencies between a motion measured by OpenGo Sensor Insoles, and the visual display on a screen.

Besides delay values, this application note also describes a procedure for measuring delays in customer-specific setups and environments, which may be of particular interest for users of the Moticon OpenGo SDKs.

2 Quick Summary

The typical propagation delays are as follows:

Data sink	Delay
Record screen of OpenGo App	0.110 s
Record screen of OpenGo Software	0.270 s
Endpoint SDK on mobile	0.087 s
Endpoint SDK via network	0.270 s
Mobile SDK	0.087 s
Insole SDK	0.087 s

Table 1 — Typical propagation delays at different OpenGo components.

3 Special Notes for SDK Users

- We here assume that the implemented software solution replacing the OpenGo App and/or OpenGo Software causes similar delays, respectively.
- The Endpoint SDK may be used on the same mobile device running the OpenGo App (localhost). Additional delays occur if the Endpoint SDK is used in a local network (or even via the Internet).

4 Typical Propagation Delay

Fig. 1 provides an overview of the components causing signal delay.

Some delay components cannot be evaluated individually with reasonable effort. Fig. 1 provides overall delays instead.

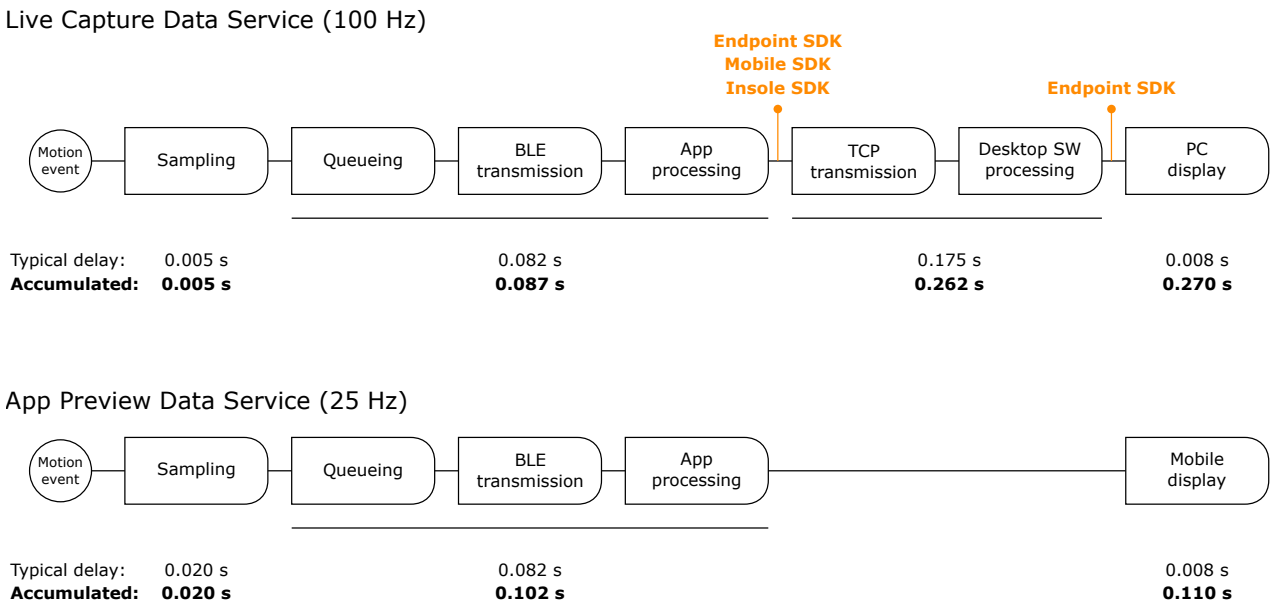

Figure 1 — Propagation delay for live capture (100 Hz) and the preview data service used by the OpenGo App (25 Hz). The numeric results are averages of three experiments carried out in a Moticon lab setup according to Sec. 6.

1. Motion event: A physical variable measured by the OpenGo Sensor Insoles, i.e. pressure, acceleration, or rotation.
2. Sampling: The sensor sampling will have missed the motion event, on average, by half of the sampling period, at maximum by a full sampling period.
3. Queueing: Queueing delays occur when a BLE data packet was not transmitted successfully on the first attempt, and due to the connection interval (see below). The queueing delay therefore depends on the number of retransmissions per packet (which is limited with BLE, before declaring the BLE devices as disconnected). In the measurements presented here, queueing delay did not play a role. The most important reasons for retransmissions are radio signal attenuation, and interference by other devices.
4. BLE transmission: The BLE transmission delay includes all delays of the BLE transmitter of the sensor insole, and the BLE receiver of the mobile device (the wireless propagation delay can be neglected). This BLE delay is largely depending on the negotiated BLE connection interval, which typically ranges between 0.0075 s and 0.040 s. The connection interval is device-specific, and may even change in the course of an ongoing transmission, in steps of at least 0.00125 s (again device-specific).
5. App processing: The app processing includes both the data handling (e.g. protocol handling on application layer), and the refresh rate of the graphics rendering architecture.
6. TCP transmission: With the standard OpenGo system, the live capture data is forwarded to a PC in the local network. The network communication link in the measurements presented here consisted of a WiFi link to a router, followed by a wired LAN link to the receiving PC.
7. PC display / Mobile display: The graphical output is finally delayed by the frame rate of the hardware screen, which was 60 fps in the experimental setup.

5 Delay Jitter

The measured delays are not fully deterministic. The biggest uncertainties are due to:

- the network delays of the TCP connection,
- the connection interval negotiated by BLE, and
- potential BLE retransmissions due to the presence of interfering devices or low link budget (large distance, obstructions between sensor insoles and mobile device).

These uncertainties depend on the specific environment. If *delay jitter* is of particular importance, we therefore recommend to carry out the procedure described in Sec. 6 multiple times, until the required statistical significance is reached.

For initial estimations without OpenGo hardware, you may consider to:

- measure network delays by whatever suitable networking tools,
- perform a desk research of the minimum supported BLE connection interval supported by the (envisaged) mobile phone model,
- assess the BLE frequency use (expected interference) in the target environment, and the feasible proximity of sensor insoles and receiver (which should be within few meters).

6 Methods

The experimental procedure for determining the delay values in this application note can easily be repeated in customer-specific setups and environments.

The procedure consisted of the following steps:

- Using a pair of OpenGo Sensor Insoles, start a live capture measurement at 100 Hz using the OpenGo App, full sensor setup.
- In the Record screen, swipe to show the acceleration widget.
- In the receiving OpenGo Software, show a line graph widget with all three acceleration axes being active.
- Line up the mobile device running the OpenGo App, the computer screen showing the OpenGo Software, and one of the sensor insoles on a table, ensuring that all are visible and in focus of a video camera filming with at least 100 fps (frames per second).
- While recording data and filming, let the heel edge of the sensor insole drop from about 2 cm above the table surface.
- Stop the data recording and video.
- Using a video software, determine the time between the heel edge of the sensor insole touching the table surface, and the very first visible peak of any of the acceleration axes in the OpenGo App bar graph widget and OpenGo Software line graph widget, respectively.

The same procedure can be applied to custom implementations using the UDP export function of the OpenGo Software, or an OpenGo SDK.

The acceleration signal was chosen because it allows for a sharper signal response than the pressure

sensors. However, since both the IMU and the pressure sensor data is transmitted in a combined data sample message, the propagation delay is equal for all data channels.

By using the visual response as reference, certain aspects of graphics generation and screen frame rates come into play. However, these can be considered in a more consistent fashion than the delays occurring with e.g. sound generation on different platforms.