Atracsys LLC
Route du Verney 20
1070 Puidoux
Switzerland

www.atracsys-measurement.com
Atracsys-Support@smith-nephew.com

Tel +41 (0)21 533 09 00

spryTrack 300

User manual

File name	sTk300_InstructionsHowToUse.pdf
Document type	Manual
Version	v4.8.0
Revision	7016d06
Pages	124
Date	8th December 2022

Contents

Glossary

Accuracy Verification Tool

The Accuracy Verification Tool (AVT) is a product designed by Atracsys. It allows the user to check the trueness of an optical tracking device. It consists of an artefact, an acquisition software, as well as a processing software running on Atracsys servers, in case a detailed report is asked for.

API

Application Programming Interface. It is a set of routines, protocols, and tools for building software applications.

Fiducial

Simplest element (sphere covered with reflective material, sticker with reflective material, reflective glass sphere, IR-LED) detected by the optical tracking system, and reconstructed as a three-dimensional point (e.g. (x, y, z) coordinates).

GUI

Graphical User Interface, this is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation.

IGS

Image-Guided Surgery. This is the general term used for any surgical procedure where the surgeon employs tracked surgical instruments in conjunction with preoperative or intraoperative images in order to guide the procedure.

Integrator

Compagny in charge of incorporating the spryTrack system component in a system. In this manual, it refers to any users of the spryTrack system.

JSON

In computing, JavaScript Object Notation or JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value).

Marker

Often also called rigid body. Mechanical structure with known geometry on which several fiducials are firmly attached. As soon as the tracking system detects and reconstructs at least three fiducials, the six degrees of freedom are computed by the system.

MITM

cyberAttack where a vicious device is connected in between the central and peripheral devices an relays and possibly alter the data.

Navigation system

A navigation system is composed of at least a tracking system and a computer on which an IGS software is running.

NFC

Set of communication protocols for communication between two device over a short distance. In the spryTrack it allows BLE paring information exchange.

OOB

Out-Of-Band communication is a way to send andor receive data outside of the main pass-band of the considered communication protocol.

Power Injector

The Power Injector is a product designed by Atracsys. It allows the user to connect a spryTrack to a computer with USB A receptacle. It combines the USB data coming from the computer and the power coming from a power supply in a single USB-C output.

Raw data

Image of a fiducial on a camera.

SDK

Software Development Kit, which is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform.

UFP

In USB Power Delivery specification, UFP or upstream facing port designates the consumer device.

USB Power Delivery

The USB Power Delivery (PD) specifies the use of certified PD aware USB cables to deliver increased power (more than 7.5 W) to devices with larger power demand. Devices can request higher currents (up 5 A) and supply voltages (up 20 V) from compliant hosts.

USB-C

USB Type-C, providing both a USB connection and power supply using only one USB cable.

Throughout this document, references to glossary entries will appear in *slanted font*.

Disclaimer of Warranties and Limitations of Liability

Atracsys has taken due care in preparing this document and the programs and data on the electronic media accompanying this document including research, development and testing. This document describes the state of Atracsys's knowledge respecting the subject matter herein at the time of its publication and may not reflect its state of knowledge at all times in the future. Atracsys has carefully reviewed this document for technical accuracy. If errors are suspected, the user should consult with Atracsys prior to proceeding. Atracsys makes no expressed or implied warranty of any kind with regard to this document or the programs and data on the electronic media accompanying this document.

Atracsys makes no representation, condition or warranty to the user or any other party with respect to the adequacy of this document or accompanying media for any particular purpose or with respect to its adequacy to produce a particular result. The user's right to recover damages caused by fault or negligence on the part of Atracsys shall be limited to the amount paid by the user to Atracsys for provision of this document. In no event shall Atracsys be liable for special, collateral, incidental, direct, indirect or consequential damages, losses, costs, charges, claims, demands or claim for lost profits, data, fees or expenses of any nature or kind.

Important notice:

The listed items in this user manual might follow different medical or non medical certifications/ standards.

Although most Atracsys devices are following medical standards, they cannot be - by their own - certified as medical devices.

For medical applications, appropriate approvals must be obtained by the integrator. Atracsys LLC disclaims all liability.

1 Important information

This documentation provides detailed information about the spryTrack system.

Information from this chapter should be read before continuing with the rest of the document. This document contains user information for the physical and software installation of the spryTrack system.

1.1 Warnings

In the whole documentation, warnings are indicated with this symbol and graphics (triangle with exclamation mark). Integrator should follow the accompanying paragraph and/or report necessary warnings in final product's instructions to avoid patient or operator injury.

Integrator should read carefully this user manual before operating the device.
Follow all the installation procedures step by step.

The integrator has to check that the device / position of the device is well suited for the application / surgery and that required regulations are respected.

Optical hazards according IEC-62471-1: Exempt group

The spryTrack device uses powerful infrared LEDs for illumination and communication. This infrared light at approximately 850 nm is not visible by human eyes. According to the standard IEC-62471-1 (hazard values are reported at a fixed distance d = 200 mm) the spryTrack is classified in the 'Exempt group'. To further reduce exposure, never watch closely and longly into the infrared LEDs around the cameras when the device is switched on. To further reduce exposure of the skin, never place any body part closer than 200 mm to the infrared LEDs.

The emission of near infrared light for measurement and communication (for example for active markers) is the intended behavior of the optical tracking system. Interference with other equipment using infrared light might be possible. As an example, interference with certain types and brands of pulse oximeters have been reported. In order to avoid the pulse oximeters to be disturbed by the optical tracking system, the pulse oximeters should be shielded either by draping or by using the designated ambient light shields.

Do not immerse the spryTrack device in water or other fluids.

Do not spill water or other fluids on the spryTrack device.

The spryTrack device is composed of metallic parts and electronic components and, even if complying with medical norms, it may interfere or be affected when used in special environments like Magnetic Resonance Imaging or devices generating X-Rays. Integrator should contact Atracsys if they intend to operate the spryTrack in such environments.

Any noisy environment such as high electromagnetic fields(e.g. MRI) might trigger a measurement error. The operability verification procedure (see Chapter 10) must be proceed to guarantee that the spryTrack is operational in the configured environment.

The spryTrack device is a precision instrument and has to be handled with greatest care. If it receives a shock, falls on any surface or is transported without adequate packaging, it can easily be damaged or decalibrated.

Since the spryTrack will be integrated in a medical system, integrator has to conduct the tests related to the electrical security and electromagnetic compatibility for the whole system.

The spryTrack must be used only according to its intended use. It must not be inserted in the human body.

The operator using the spryTrack must be correctly trained by the integrator, to respect at least, all the requirements of this manual.

The spryTrack must be used only according to its intended use. The spryTrack should only be use with the Atracsys library provided in spryTrack SDK.

1.2 Caution

In the whole documentation, cautions are indicated with this symbol and graphics. The information from the accompanying paragraph must be followed to avoid damaging the equipment.

1.3 Safety signs and symbols

Various symbols are used in this manual, the user's manual and on the product itself to ensure correct usage, to prevent danger to the user and others, and to prevent property damage. The meanings of these symbols are described below. It is important that these descriptions are read thoroughly and the contents is fully understood.

The entire user manual must be read before operating the spryTrack system.

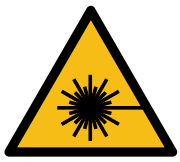
IP20

The spryTrack device degree of protection against dust and fluids (IEC standard 60529).
First digit: Solid particle protection
Level 2: Protected from touch by fingers and objects greater than 12 mm.
Second digit: Liquid ingress protection
Level 0, Not protected from liquids.

A parcel containing a spryTrack device must be handled with care.

A parcel containing a spryTrack device must be transported and stored between the indicated temperatures. See Section 5.2.

The spryTrack device must not be disposed of in the trash, see Section 14.4.



LASER RADIATION
Do not stare into beam. Class 2 Laser Product
$\lambda = 400 - 700\text{nm}$ $P_o \leq 1\,\text{mW}$

LASER RADIATION for structured light
Do not stare into beam. Class 1 Laser Product
$\lambda = 850\text{nm}(nominal)$ $P_o \leq 300\,\text{mW}$

1.4 Device label

This section details the information printed on the label of the device. An example is provided on Figure 1.2. The label gives the serial number of the device, its version and the date of manufacture (format yyyy-mm-dd). The serial number is encoded in the data matrix barcode using ASCII character encoding, e.g. '`0x0000000000000000`'.

The label contains the certification ID of the BLE module. The BLE MAC address is provided, it can be omitted at the request of the integrator.

To fulfill requirements 5.1.1 of IEC 60601-1-2:2007 (ed3.0), in case the spryTrack is used in BLE mode, integrator has to marked the equipment with symbol IEC 60417-5140 for non-ionizing radiation.

Figure 1.1: IEC 60417-5140 non-ionizing radiation symbol.

The spryTrack is powered by *USB Power Delivery* . All the accepted profiles are listed on the label.

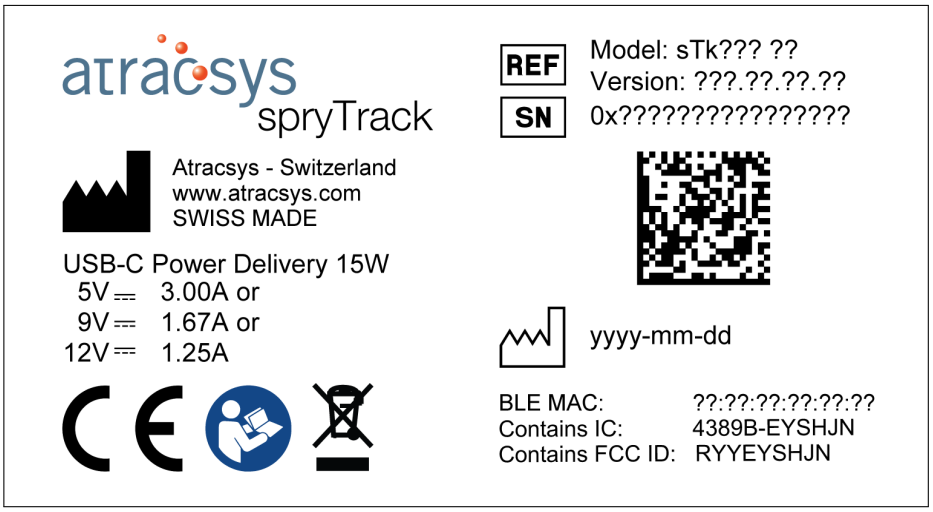


Figure 1.2: Blank example of a spryTrack label.

1.4.1 Power Injector label

This section details the information printed on the label of the *Power Injector*. An example is provided on Figure 1.3. The label gives the serial number of the power injector, its version and the date of manufacture (format yyyy-mm-dd).

The power injector is powered by 5 V to 6 V,3 A *USB-C* power supply and outputs *USB Power Delivery* profile 5 V, 3 A to the spryTrack

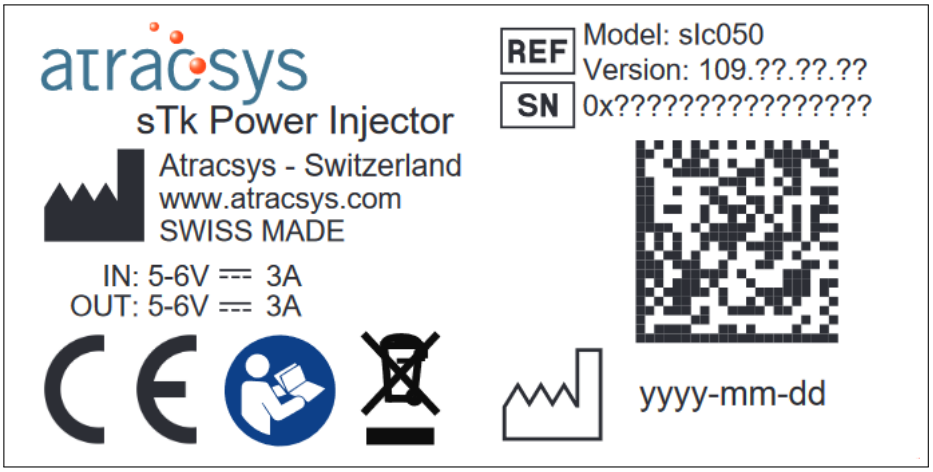


Figure 1.3: Blank example of a *Power Injector* label.

1.5 Disclaimers

The entire user manual must be read before operating the spryTrack system.

1.6 Contact information

For any question regarding the content of this guide or the operation of this product, please contact us:

Atracsys LLC
Route du Verney 20
1070 Puidoux
Switzerland

www.atracsys-measurement.com
Atracsys-Support@smith-nephew.com

Tel +41 (0)21 533 09 00

1.7 Updates

The latest version of this document can be downloaded from Atracsys website. The address, as well as the required username and password are provided by Atracsys, please refer to the 'Information for user' document.

2 Introduction

The spryTrack is a compact and mobile optical pose tracking system at the edge of the technology with no compromises on speed nor precision.

The extremely compact spryTrack is composed of two cameras designed to detect and track *fiducials* (reflective spheres, disks and/or IR-LEDs) with high precision in real time video streams. Triangulation enables retrieving 3D position of each *fiducial* with sub-millimetric accuracy. When several *fiducials* are affixed to a *marker*, its pose (orientation and position) is calculated with 6 degrees of freedom $(x, y, z, \alpha, \beta, \gamma)$. The spryTrack has the ability to provide 3D positions of the *fiducials*, and/or poses of the *markers*.

The spryTrack offers both a USB Type-C (for power and/or data) and Bluetooth (BLE) connections allowing a wireless link to a tablet on which the navigation application runs. A PC is no more needed. An optional battery pack enables complete mobility.

The spryTrack has two operations mode USB and BLE.

The SDK allows access to data at different stages of processing, starting from raw images, individual 3D positions of *fiducials*, and up to the pose of *markers*. The SDK also provides multi-level fault checking. This makes it possible to access error information in real-time at any processing stage: *fiducial* occlusion level, stereo de-calibration, marker registration error and more.

The spryTrack can be customized to fit your requirements (e.g. precision level, acquisition speed, working volume, extensions). Moreover, the system is compatible with existing passive image-guided surgical (*IGS*) tools that are widely used in the medical field.

The spryTrack system is one of the components of an *IGS* tool. The final (i.e. surgical) application will depend on the requirements of the integrator.

The spryTrack measures distances and angles between *markers* fixed on the patient, as well as *markers* fixed on the surgical tools. Data generated by the spryTrack are typically used by the *IGS* system to display information such as distances, angles, trajectories, position of surgical tools within preoperative images, etc. The entire *IGS* system (including the spryTrack) helps surgeons to take decisions during the procedure.

The spryTrack is not intended to make surgical decisions by itself or to replace surgical actions.

This document presents how to install and use the spryTrack system. It is, as well as the current spryTrack software, continuously improved and may differ from one version to another.

For any problem, question or suggestion do not hesitate to contact Atracsys either by email or by phone (see Section 1.6).

2.1 Device expected service life

The expected service life of the spryTrack is seven years based on five hours of use per working day. However, the shock sensor battery should be recharged every 6 months.

2.2 Device specifications

The specifications of the spryTrack 300 are presented in Table 2.1. The spryTrack 300 measurement working volume are reprensent in Figure 2.1. The measurement working volume is the volume on which the camera is calibrated and the accuracy is verified by Atracsys. It is a little smaller than the actual visible volume and is limited between 400mm and 2400mm in longitudinal distance with spherical back plane.
The Typical fiducial localisation error at calibration time are:
0.14 mm RMS / 0.27 mm 95% CI[a] up to 1.4 m
0.20 mm RMS / 0.41 mm 95% CI[a] up to 2.4 m.

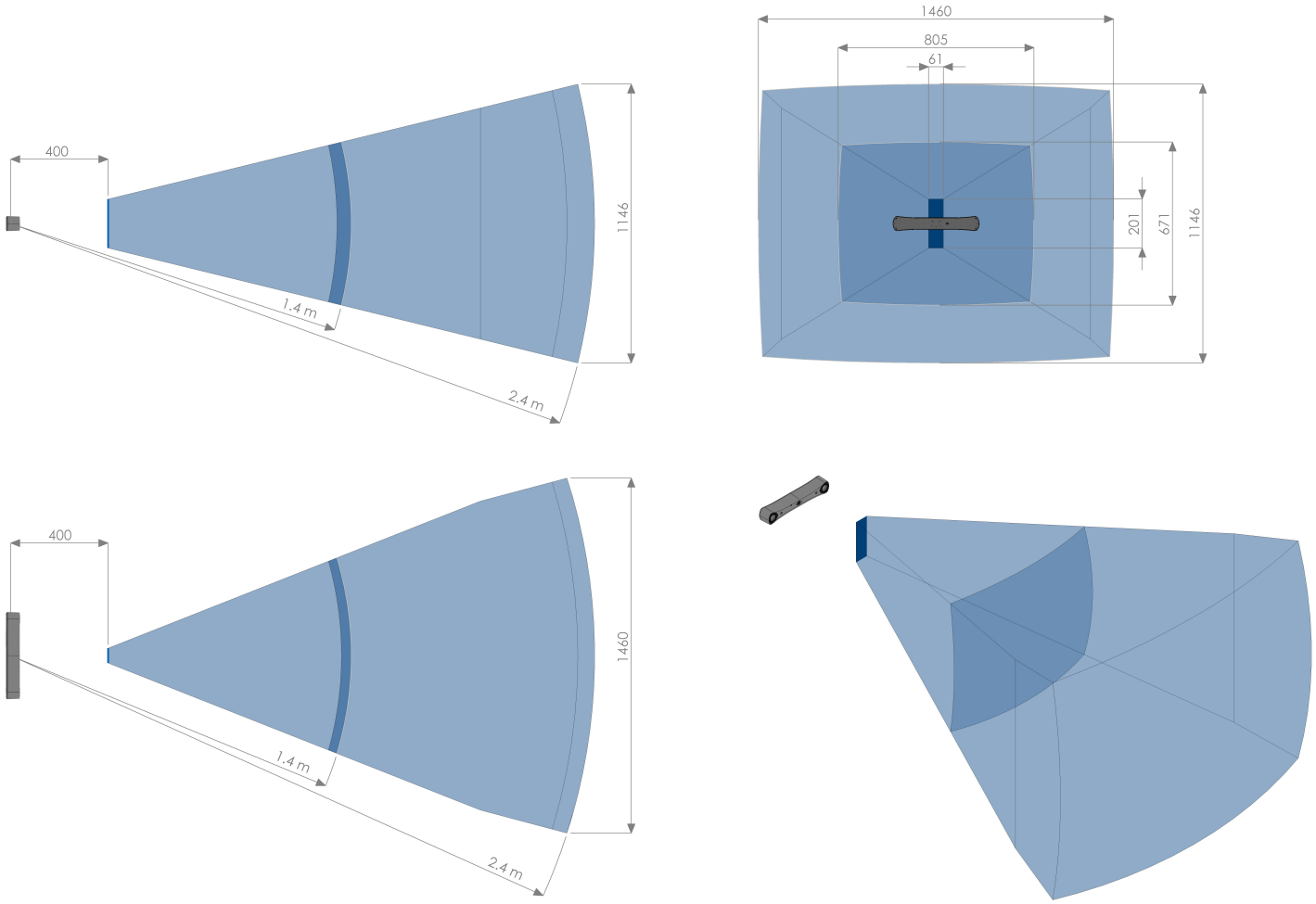


Figure 2.1: sTk300 Measurement working volume specification.
Cross-sections of the measurement volume at depths of 400 mm (front plane, in dark blue), 1400 mm (in mid-tone blue) and 2400 mm (in light blue).

Device	spryTrack 300
Model	sTk300
Dimensions (L × W × H)	356.5 mm × 60.5 mm × 55 mm
Weight	1073 g
Typical fiducial localisation error at calibration time	0.14 mm RMS / 0.27 mm 95% CI[a] up to 1.4 m 0.20 mm RMS / 0.41 mm 95% CI[a] up to 2.4 m (Measurement working volume)
Accuracy	*Accuracy Verification Tool (AVT)* distance RMS error < 0.6 mm, measured in the Measurement working volume.
Measurement working volume	Projections drawn in Figure 2.1.
Infrared illumination	∼ 850 nm
Measurement rate	54 Hz[b], native measurement rate (no upscaling)
Image exposition time	Active 0 µs to 16 383 µs Passive 0 µs to 250 µs
Latency	∼ 25 ms[c]
Wired interface (to PC)	USB Type C - USB 2.0[d], USB 3.0 SuperSpeed
Wireless interface (to tablet)	Bluetooth Low Energy 5.0
Max. simultaneous *markers*	16 *markers* over USB, 4 *markers* over Bluetooth Low Energy
Max. *fiducials* per *marker*	5 spheres / discs / IR-LEDs per *marker*
SDK	C/C++ (shared library), Python
Operating systems	Windows / Linux / Android (see Section 3.1)
Power requirements	*USB Power Delivery* Max 15 W. Power profiles: 5 V,3 A; 9 V,1.67 A; 12 V,1.25 A
Pollution degree	2
Overvoltage category	I
Acoustic energy	< 80 dB
Laser	Class 2
Standards	- Basic safety and essential performance: IEC 60601-1:2005+A2:2020, Ed 3.2 - Electromagnetic disturbances: IEC 60601-1-2:2014+A1:2020, Ed 4.1 - Photobiological safety of lamps: IEC 62471:2006, Ed 1.0 - Safety of laser products: IEC 60825-1:2014, Ed 3.0 - Medical device software: IEC 62304:2006+A1:2015, Ed 1.1

[a] Based on a single fiducial stepped uniformly throughout the measurement working volume at 20 °C.
[b] Depending on user's computer, USB connection and numbers of blobs detected in the pictures.
[c] Tested with a USB connection and in the case of typical IR images with 4 markers including 4 fiducials in the center of the Working Volume and without interference
[d] Performances may be degraded. Please contact Atracsys if you plan to use this interface.

Table 2.1: spryTrack 300 specifications

3 Description of the spryTrack system

The spryTrack family of Optical Tracking Systems is a flexible, real-time tracking technology based on a stereo camera approach. The spryTrack is based on measuring the 3D position of *fiducials* to provide real-time orientation and position of each *marker*.
For shock and temperature stability, the cameras inside the spryTrack housing are floating (see Figure 3.1) and therefore the spryTrack is not designed to measure absolute positions (positions and rotations in respect to the spryTrack housing / the fixation base). The spryTrack is designed to provide accurate relative measurements. 'Relative measurements' means that position and rotation from one respectively several markers to each other are computed.

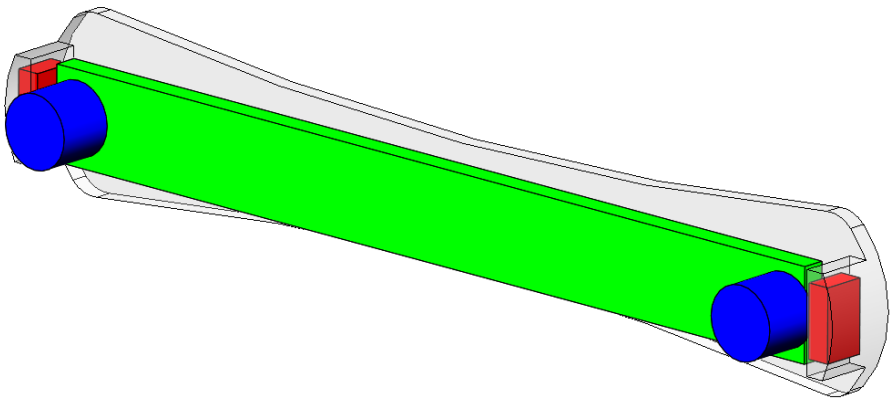


Figure 3.1: Sketch of the spryTrack device. The cameras (in blue) are mounted on a rigid rod (in green). This rod is fixed to the 'shell' (grey) by rubber elements (red).

3.1 Hardware requirements

The minimum host PC requirement to use the spryTrack system in USB mode are:

- Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz
- 4 GB DDR3 RAM:
- 200 MB disc space;
- Windows 10 (64 bits supported);
- Linux (64 bits supported), gcc 5.4 or clang 3.8.
- The Linux Kernel support for USB3 started at version 2.6.31.

The minimum host requirement to use the spryTrack system in BLE mode are:

- Linux Kernel greater than 3.4
- BLE 5.1

3.2 The spryTrack device

Here below is a simple description of its operation in USB mode:

1. The spryTrack illuminators emit infrared (IR) light, like the flash of a regular camera;
2. The IR light reflects on passive *markers* or triggers active *markers* which then emit IR light;
3. The spryTrack detects the reflected (or emitted) IR light, if the option ”image sending” is enabled, the spryTrack transmits the information to the host computer along with status information;
4. The host computer or the spryTrack (depending if the option ”onboard processing” is enabled) extracts the position of each detected spot, computes the 3D position of each detected sources, then tries to match them to known *marker* geometries;
5. If the option ”image sending” is enabled, the information is forwarded to the user, with status information on each piece of data, allowing the user to do additional processing (display, filtering, collection, etc.).

The spryTrack device can track two types of *markers*: passive and active ones. More detailed explanations about those two *marker* types are given in Section 7.7.
More detailed explanations about spryTrack option are given in Section 8.1.1.

The spryTrack (IR devices only)camera is equipped with infrared interferometric bandpass filter. As the angle of incidence (AOI) has an impact on this type of filter (blue shift effect), for all AOI a bandpass interval of 840 nm to 900 nm is guaranteed. Light from a source in a near interval 740 nm to 1000 nm might be seen but the AOI will have an impact that should be evaluate by the user. Wavelengths far from this interval (outside the 740 nm to 1000 nm interval) will cut by the IR filer. The spryTrack (IR devices only) image sensor has a quantum efficiency response of $\sim 20\%$ at 840 nm and $\sim 12\%$ at 900 nm with an almost linear behavior between these two values.

On Figure 3.2 is presented the front view of the spryTrack. It should be noted that ‘Left’ and ‘Right’ always refer to the vision from the spryTrack.

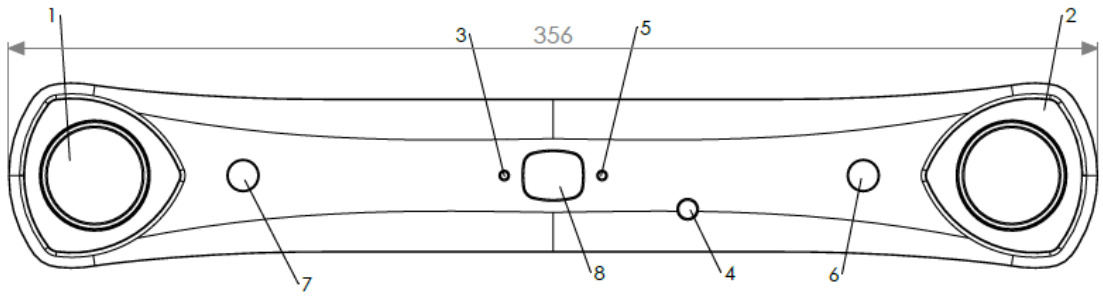


Figure 3.2: The spryTrack front view, showing: (1) Right camera; (2) Left IR-LEDs ring; (3) User Status LED, refer to Section 3.2.3 for more information; (4) IR receiver; (5) Device Status LED, refer to Section 3.2.3 for more information; (6) Right pointing laser; (7) Left pointing laser; (8) Point projector.

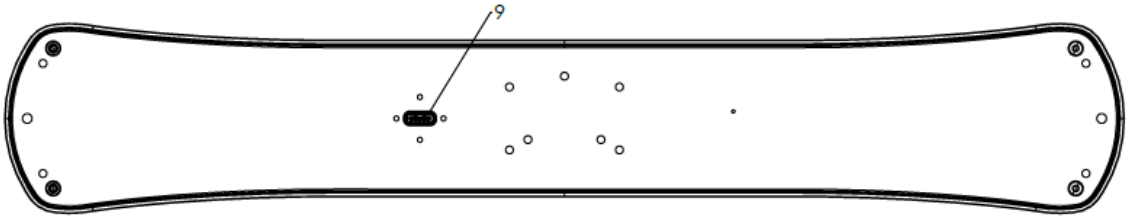


Figure 3.3: The spryTrack backplane, showing: (9) USB 3.0 Type-C connector, providing the power and the transmission of data between the spryTrack and a computer.

The backplane of the spryTrack is presented on Figure 3.3. The USB interface provides both communication and power. The device is switched on / off by deplugging the USB cable.

3.2.1 Power input

The used power supply must be a limited power source. Any used power supply must not be able to deliver more than 100 W.

The power supply has to be compliant with IEC60601-1:11 standard or equivalent.

The spryTrack is a USB Type-C consumer device that sinks current (*UFP*) respecting the USB-IF standard. For more information on the USB Type-C power delivery, please check USB Type-C documentation. The spryTrack needs at least 15 W.

In USB mode, the spryTrack can either be plugged directly to a computer that manages USB-PD protocol, or it can also be used with a *Power Injector*. The *Power Injector* takes as input a power supply via USB and a USB 3.0 PC host, and outputs a complete USB-C: fused power from the power supply and USB3.0 from the host PC. The *Power Injector*'s main goal is to split power and USB 3.0 sources. The *Power Injector*

also increase the distance between the spryTrack and the Host PC. See section 5.5 and figures 5.2 and 5.4 for more information on how to connect the *Power Injector* to the spryTrack device.
The list of *USB-C Power Injector* successfully tested by Atracsys is found in Table 3.1.

While using the spryTrack in BLE mode, the input power could also be provided directly with the provide AC adaptor. The same power supply can be used to power the spryTrack in BLE mode or the *Power Injector*. Atracsys strongly recommends to use the provided power supply.

If you intend to use the spryTrack system in USB mode with the *Power Injector*, there is a known defect making a USB 2.0 instead of a USB 3.0 speed connection. While *USB-C* is normally flippable, due to this defect, one side of the USB-C connected in the USB port for the computer of the *Power Injector* the speed can be limited to USB 2.0. Because the issue happens only for one side, if it was to occur, flip the *USB-C* connector in the USB port for the computer of the *Power Injector*. This side the spryTrack should have a USB 3.0 connection speed.

Product	Version
Atracsys *Power Injector*	slc050 109.00.02.00

Table 3.1: Atracsys *USB-C Power Injector* devices.

Manufacturer	Reference
Globtek	GTM96180-1507-2.0 USB-C

Table 3.2: Atracsys recommended AC/DC power supply.

Manufacturer	Reference
Globtek	GTM96181-18PD-PPS-Q,Ua 5/9/15V max. 3A, 18W
Globtek	GTM96181-36PD-PPS-Q, Ua 5/9/15/20V max. 3A, 36W
SL Power Electronics	ME20A0596B02, Ua 5V max. 3A, 15W, USB-C
Egston	E2CFMW3 24W LV6 USB-C, Ua 5V max. 3A, 15W, USB-C

Table 3.3: AC/DC medical power supply tested by Atracsys

Others powers supply that have been functionally tested by Atracsys are listed in Table 3.3.
In addition Atracsys may provide a bare USB-C Powersupply cable on request Table 3.4.
This cable allows powering the spryTrack or the *Power Injector* from a 5 V DC source. This is the same cable that is installed in the provided power supply Table 3.2.

Electromagnetic compatibility tests (see Chapter 13) of the spryTrack has been performed with the *Power Injector* and the power supply listed in Table 3.2. For any others power supply, including power supply listed in Table 3.3 EMC and electrical safety compliance (see Chapter 13) has not been performed and are not guaranteed by Atracsys.

Manufacturer	Reference
Globtek	WR9QA3000USB-CIMR68 USB-C

Table 3.4: Atracsys USB-C Powersupply bare cable

3.2.2 Cables

Atracsys does supply cables with the spryTrack system upon request. Available cables are listed in Table 3.5. Atracsys highly recommends to use cables listed in Table 3.5. Otherwise, the USB Type-C (*USB-C*) cables should be chosen carefully as it should support the current drawn by the spryTrack system. In case an Atracsys *Power Injector* is used, two USB cables are needed (one *USB-C* cable from the host PC to the Atracsys *Power Injector* and one *USB-C* to *USB-C* cable from the Atracsys *Power Injector* to the spryTrack device). The Atracsys *Power Injector* also needs a main power supply.

USB Type-C is still being standardized, please check USB Type-C documentation.

Cable description	Manufacturer	Reference
USB-C to USB-C Cable 2m with lock-screw	JAE	DX07550B20K19510
USB-C to USB-A 3.0 Cable 1.83m	Tripp Lite	U428-006

Table 3.5: Atracsys recommended *USB-C* cables for *Power Injector* and spryTrack system.

Electromagnetic compatibiliy tests (see Chapter 13) of the spryTrack has been performed with the *Power Injector* and cables listed in Table 3.5. For any others USB cables, EMC and electrical safety compliance (see Chapter 13) is not guaranteed by Atracsys.

3.2.3 Status LEDs

The status LEDs colour and behaviour is used to indicate the status of the spryTrack device. When the spryTrack device is switched on, the boot sequence starts with tests, during which the LEDs are switched on and off, with various colors. This lasts for about 20 s to 30 s, then the LED is used to indicate the status of the device. The various states are explained on Table 3.6.

LED	Colour	Behaviour	Description
Device Status LED	Green	Blinking	Device is warming up, initialisation procedure.
	Green	Solid	Device is ready, and Bluetooth is in classic advertising mode.
	Blue	Solid	Device is ready, and Bluetooth is in *Near-Field-Communication* mode.
	Red	Solid/Blinking	Indicates a fatal error, please contact Atracsys.
User Status LED	–	–	User Status LED is available for the user to indicate personal info. The colour and flashing frequency can be set through the SDK.

Table 3.6: Description of the LEDs patterns

3.3 The spryTrack software development kit

The spryTrack software development kit (*SDK*) is used to retrieve the measurements from the spryTrack on the PC and to modify settings on the spryTrack. It is completely documented in an annex document [1].

3.4 Authorised modifications

Strictly no modifications are allowed on the spryTrack nor on the spryTrack SDK. The spryTrack device shall not be opened. If repairs or modifications are needed, please either contact or send the device back to Atracsys (Section 1.6).

4 Shock sensor

The spryTrack device is equipped with a shock detection circuitry, hereafter called shock sensor. The shock sensor is a very low power, battery powered electronic unit designed by Atracsys based on a microcontroller (μC). The unit is composed of three accelerometers, a temperature sensor, a real time clock (RTC) and a non-volatile memory. Two of the three accelerometers are continuously monitoring shocks that might happen to the tracking system and the third one is providing the device orientation. Additionally, the shock sensor also records information about the use of the tracking system, such as the number of boot or the usage time.

The primary purpose of the shock sensor is to detect and record shocks encountered by the tracking system above a certain threshold, actually set to $100g$, where $g = 9.81\,\mathrm{m\,s^{-2}}$ is the average Earth gravitational constant. Thanks to the built-in rechargeable battery, theses shocks are recorded even when the tracking system is not powered. Each shock event is timestamped by the real-time clock. If such a shock event over the threshold has happened, the user is informed over the SDK if the corresponding option is activated.

The battery is automatically recharged each time the spryTrack device is powered. Therefore, in a regular usage scenario, the battery is always (almost) fully charged. A completely empty battery is fully recharged in 24 h. The system runs autonomously for approximately 6 months without recharging the battery. In case the battery runs out of power, the shock sensor will be safely stopped and, thanks to the non-volatile memory, the recorded data stays saved. Shocks that might happen during the time where the shock sensor has an empty battery are of course not logged anymore. The user however is informed, if the battery runs out and therefore non-recorded shocks might have happened. In such an empty battery event, to validate the accuracy of the tracking system, Atracsys recommends to run an AVT analysis (see Chapter 12).

If not used, the spryTrack device should be recharged for 24 h every 6 months.

The shock report can be downloaded, the tool that retrieve the shock sensor data is described in section 8.7.4.

5 Hardware installation

In this chapter is presented how to properly set up a spryTrack device.

When unpacking the received spryTrack, it must be checked that no item is missing, and no item is damaged. The invoice or shipping paper defines the list of items in the shipment. The Atracsys after sale service should be contacted in case of a missing or damaged item.

5.1 Operating environment requirements

The spryTrack can be used covered: for instance with a sterile cover. However, this may change the operating temperature of the device and may change the optical path, both resulting in a change in the calibration parameters, causing a degradation of the device accuracy. However, the spryTrack should not be located in the sterile field.

The spryTrack must not be used if a hot air outlet is present between the spryTrack and the *markers*. The refraction index changes of the hot and cold air interfaces could cause wrong measurements.

The spryTrack has to be placed horizontal. An orientation other than horizontal (e.g. vertical) can detoriate precision due to thermal gradients inside the device.

To allow the spryTrack to deliver measurements, the *markers* must be in the spryTrack working volume during the whole navigation process.

When powered on, the spryTrack must be installed such that a minimum separation distance of 20 mm is maintained between the device and all persons/body parts at all times.

The spryTrack cameras need a high contrast (in the IR spectrum, $\sim$ 850 nm) to operate properly. Reflective surfaces and polluting lights (sunlight, lamps with a strong IR component around 850 nm, etc.) must be avoided. The SDK enables the reception of the images as seen by the cameras. In these images, polluting sources can be identified. During operation, potential sources of sunlight (or equivalent) must be eliminated. The usage of active fiducials (IR-LEDs) can help to reduce reflections.

The spryTrack products are intended for indoor use in professional healthcare facility environments like physician offices, dental offices, clinics, limited care facilities, freestanding surgical centers, freestanding birthing centers, multiple treatment facilities, hospitals (emergency rooms, patient rooms, intensive care, surgery rooms) except for:

- near active HF (high frequency) SURGICAL EQUIPMENT,
- inside RF (radio frequency) shielded room of an ME SYSTEM for magnetic resonance imaging, where the intensity of EM DISTURBANCES is high
- in ambulances.

The calibration of the device is performed at 20 °C. The best accuracy and precision performances are obtained when the spryTrack is operated in a room at 20 °C, once thermal equilibrium is reached. The spryTrack should be used in the conditions listed in Table 5.1.

Environmental parameter	Range
Temperature	18 °C to 25 °C
Atmospheric pressure	70 kPa to 106 kPa
Relative humidity	20 % to 80 % non condensing
Altitude	−450 m to 3000 m

Table 5.1: Recommended environmental conditions.

5.2 Transportation and storage environmental conditions

The integrator of the spryTrack system must take care of the mechanical stability of the spryTrack at all times (when the spryTrack is in use but also when it is not powered). Effectively, the calibration of the device may be invalidated if the device suffered from shocks, drops and vibrations. The spryTrack should therefore be transported and stored in an adequate transportation / storage package which minimises the probability of shocks, drops and vibrations and absorbs mechanical shocks and stress as good as possible. If the spryTrack device was stored at extreme environmental conditions, a reasonable amount of time must be waited before using it, so that the operating environmental conditions are reached.

Environmental parameter	Range
Temperature	−29 °C to 60 °C
Atmospheric pressure	50 kPa to 106 kPa
Relative humidity	10 % to 90 % non-condensing

Table 5.2: Recommended short time (< 1 week) storage and transportation conditions.

The spryTrack has been designed to be stored and transported in the conditions listed in Table 5.2 and Table 5.3. Note that during both transport and storage, special handling measures have to be taken:

- keep away from direct sunlight;
- keep away from strong heat or cold;

Environmental parameter	Range
Temperature	−10 °C to 55 °C
Atmospheric pressure	50 kPa to 106 kPa
Relative humidity	10 % to 90 % non-condensing

Table 5.3: Recommended long-term (1 week – 6 months) storage and transportation conditions.

- keep safe from shocks, drops and vibrations;
- keep away from fluids;
- a 'fragile' marking must be present on the package.

5.3 Mounting the spryTrack device

The integrator has to perform tests related to instability hazards §9.4 of IEC 60601-1.

The integrator of the spryTrack system must take care of the mechanical stability of the spryTrack at all times (when the spryTrack is in use but also when it is not powered). Effectively, the calibration of the device may be invalidated if the device suffered from shocks, drop and strong vibrations. The spryTrack must therefore be firmly fixed to avoid any fall and should be mounted on a structure which minimises the probability of shocks, drops and vibrations.

When installed on a tripod or on a cart, special care has to be taken to avoid tipping over. The shocks, drops and vibrations have also to be avoided when the equipment on which the spryTrack is installed is moved like for example on a cart that passes through doors, is moved over a threshold or is put in the parking position.

Firm mounting can be achieved by using four (4) M3 screws (max depth 3 mm) or three (3) M3 screws in the center (max depth 3 mm). More explanations are provided on Figure 5.1

5.4 Connection of the spryTrack device

The spryTrack uses a USB 3.0 Super Speed interface to communicate with the PC or Bluetooth Low Energy to communicate with a wireless device (such as Android, iOS tablets). The spryTrack must be properly powered (see Section 3.2.1).

Using the spryTrack through a USB2.0 connection is not recommanded, please contact Atracsys if you want to use this configuration.

When using the spryTrack through a USB2.0 connection, it is highly recommended to *enable the embedded processing* and to *toggle off the images sending*. Otherwise, the framerate may be degraded.

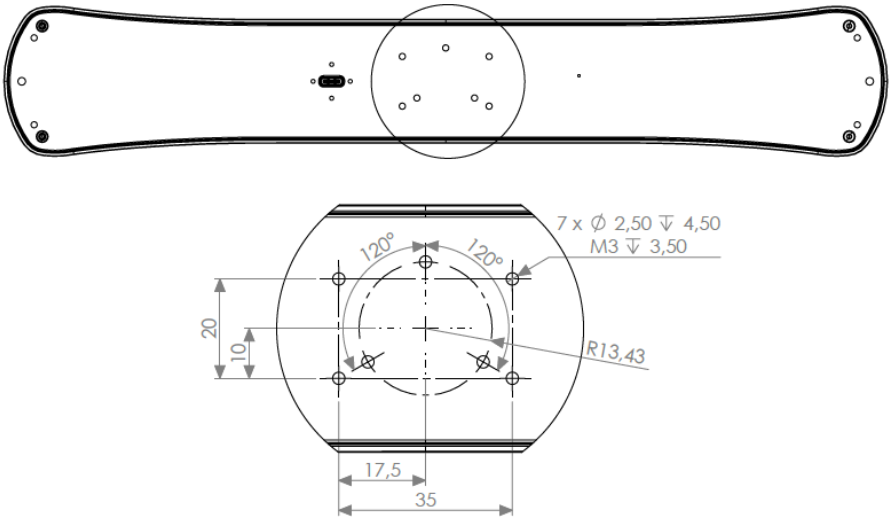


Figure 5.1: The spryTrack screws fixation.

If the spryTrack is used with a *Power Injector*, due to a driver issue the first time the device is booting it may be detected only in USB2.0. In this case, please unplug and replug the USB cable of the *Power Injector* from the host PC.

In order to avoid USB sniffing attacks, the spryTrack must not be used through a USB hub.

5.5 Connecting and powering the spryTrack

This section is an extract of the document 'spryTrack electrical safety concept'. Upon request, a complete copy can be provided.

5.5.1 The spryTrack

Even though no patient contact is necessary for the spryTrack in normal operation, the spryTrack has to be considered as Type B applied part since the spryTrack can potentially be placed within the patient environment (up to 1.5 m from the patient). Also indirect connection is possible if for example the surgeon touches the patient at the same time that they are touching the spryTrack to adjust its position.

A Type B applied part according to the medical standard 60601-1 has to comply with:

- Requirements of the 60601-1 to provide **protection against electric shock**;
- Requirements of the 60601-1 regarding **touch current / enclosure leakage current**;
- Requirements of the 60601-1 regarding **patient leakage current**;

- Requirements of the 60601-1 regarding **patient auxiliary current**.

Patient leakage current and patient auxiliary current are not applicable since the spryTrack has no patient connection (Type B applied part); it is only *considered* as Type B applied part.

Therefore only the two following requirements have to be respected:

- Requirements of the 60601-1 to provide **protection against electric shock**;
- Requirements of the 60601-1 regarding **touch current / enclosure leakage current**.

spryTrack protection against electric shock

The spryTrack is certified following the medical standard 60601-1. Following this standard, two Means of Patient Protection (2 MOPP) are necessary in order to guarantee the patient's safety. The spryTrack systems provide no isolation between the USB connection (working voltage of up to 12 VDC for *USB Power Delivery*) and the housing of the spryTrack. Therefore, the power provider (power supply or computer) must guarantee the two Means of Patient Protection.

spryTrack touch current / enclosure leakage current

The following measure have been extracted form the spryTrack IEC60601-1 tests report. Only the maximun measured value (the spryTrack has been tested in the two functionning mode (BLE or USB via the *Power Injector*, see section Section 3.2.1) are listed. Futher details on request.

Condition	Standard leakage current measurement		Leakage current measurement non-frequency-weighted device	
	Maximum measurement	Allowable value	Maximum measurement	Allowable value
Normal condition	60 µA	100 µA	70 µA	10 mA
Single fault condition	93 µA	500 µA	94 µA	10 mA

5.5.2 spryTrack in a ME system

The *Integrator* has to discuss and verify the spryTrack integration in a Medical Electrical system (ME system) with a certification body. This section of the manual is provided as information only.

If the spryTrack is used as a component in a Medical Electrical system (ME system), the whole system has to respect the 60601-1 standard in order to guarantee the safety of the operator and the patient:

- Requirements of the 60601-1 to provide **protection against electric shock**;
- Requirements of the 60601-1 regarding **touch current / enclosure leakage current**.

The standard for medical electrical equipment medical covers ME systems that are built as a combination of – not necessarily medically certified – components. The safety of the whole ME system can therefore be ensured when respecting the following points:

- A 60601 component can be combined with components that respects relevant safety standards but not necessarily the medical 60601 standard
- The *total* touch current of the ME system has still to respect the 60601 standard (100 µA in NORMAL CONDITION and 500 µA in SINGLE FAULT CONDITION).

ME system protection against electric shock

Multiple use cases respecting requirements of the 60601-1 to provide protection against electric shock are possible. Figure 5.2, 5.3, 5.4 and 5.5 show the different use cases of a spryTrack connected through USB to a computer.

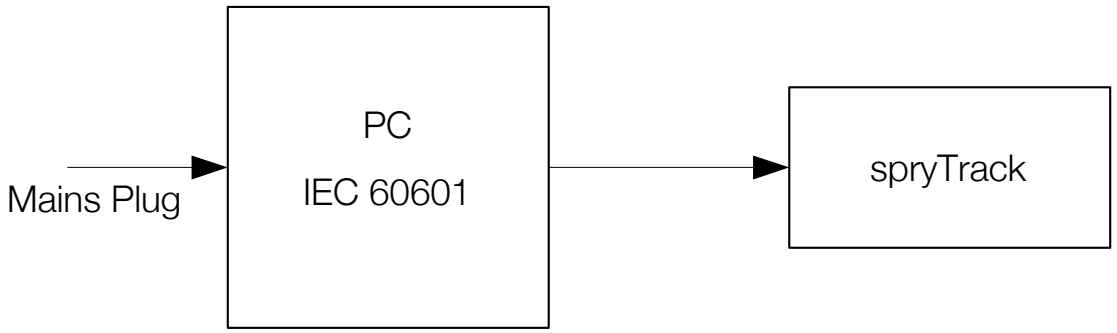


Figure 5.2: The spryTrack can be directly connected to a 60601 computer.

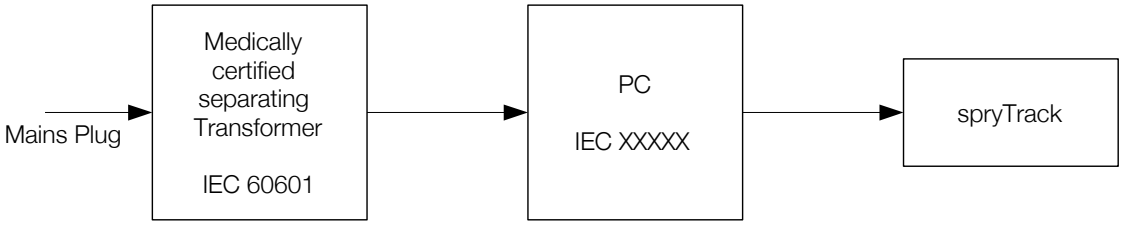


Figure 5.3: The spryTrack is direclty connected to a non 60601 computer. To fulfill the requirement, a medically certified separating transformer is needed.

Figure 5.6 and 5.7 show the two use cases possible of a spryTrack connected to a computer/tablet through Bluetooth.

ME system touch current / enclosure leakage current

It is mandatory that the *total ME system* respects the limits for the touch current.

60601-1 states (non-exhaustive list):

- In NORMAL CONDITION, the TOUCH CURRENT from or between parts of the ME SYSTEM within the PATIENT ENVIRONMENT shall not exceed 100 µA.
- In the event of the interruption of any non-PERMANENTLY INSTALLED PROTECTIVE EARTH CONDUCTOR, the TOUCH CURRENT from or between parts of an ME SYSTEM within the PATIENT ENVIRONMENT shall not exceed 500 µA.

There is no means to calculate or estimate the total touch current; it has to be measured. Even if all individual components (medically certified and non-medically certified) respect the limits of 100 µA in NC and 500 µA in SFC, the combination of several components is most likely to exceed the limits. In this case, a medically certified separating transformer between the mains and the ME system will guarantee the total

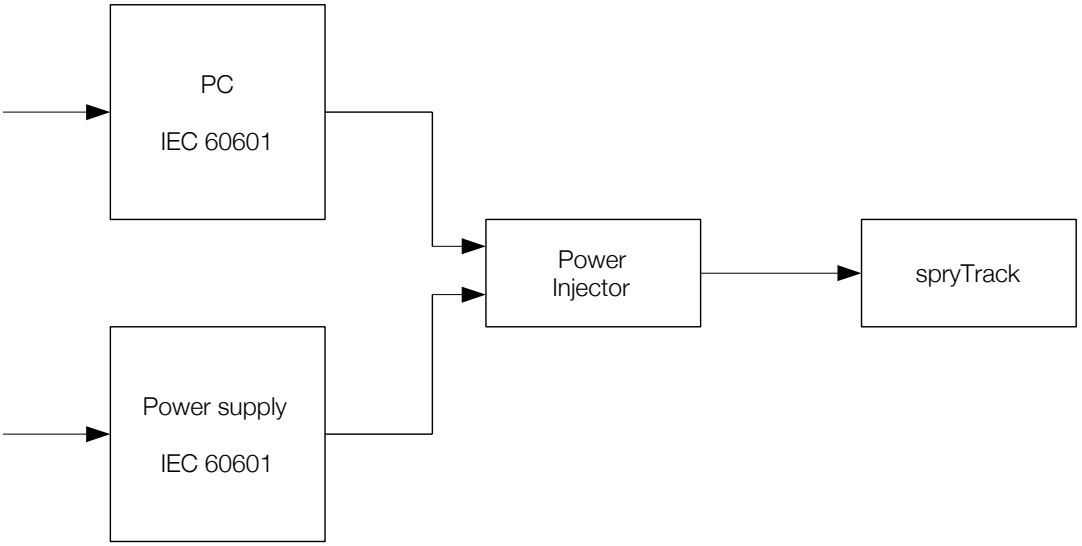


Figure 5.4: The spryTrack is connected to 60601 computer through the *Power Injector*. A 60601 power supply is needed for the power injector. Verify total patient leakage current.

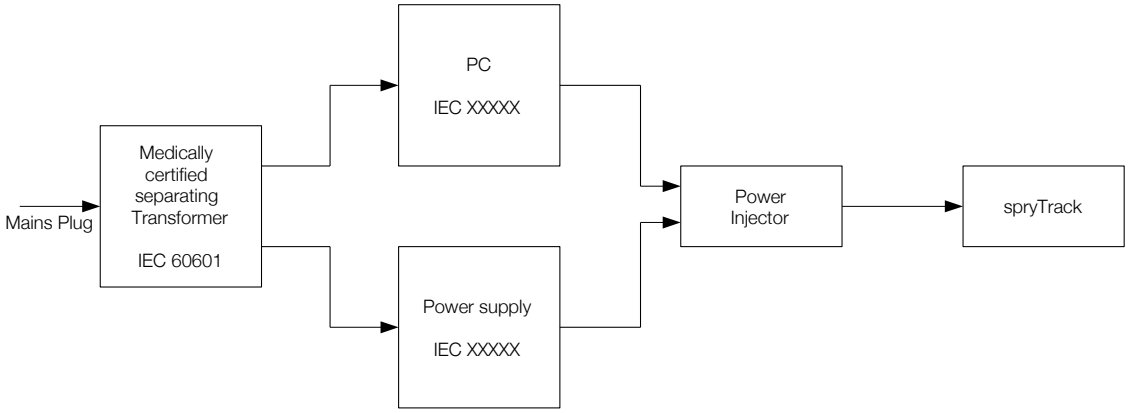


Figure 5.5: The spryTrack is connected to a non 60601 computer through the *Power Injector*. A medically certified separating transformer is needed.

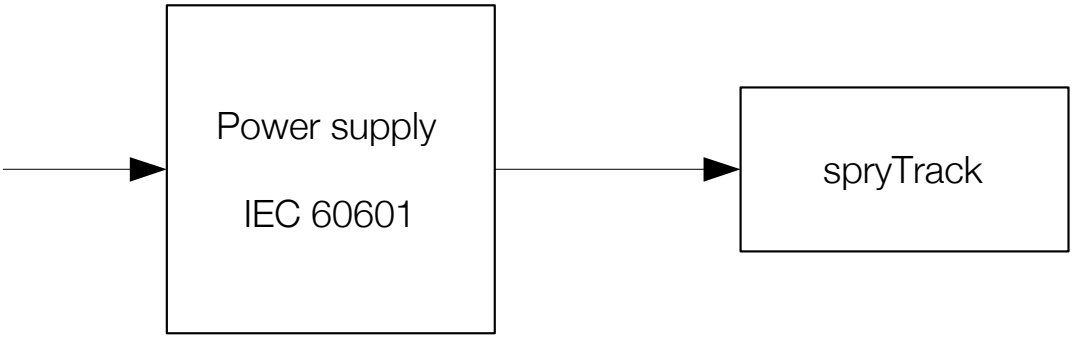


Figure 5.6: The spryTrack can be powered by a 60601 power supply.

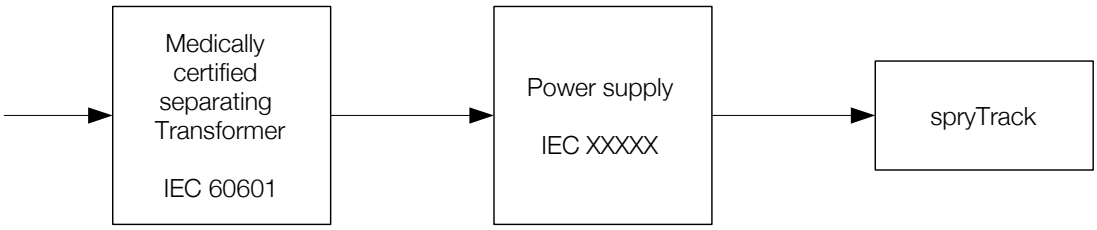


Figure 5.7: The spryTrack is powered by a non 60601 power supply.A medically certified separating transformer is needed.

touch current to stay under the required limits. Additionally, the medically certified separating transformer also increases the means of protection regarding electric shock for the whole ME system and also for the individual components.

5.5.3 Annex: Explanation of applied part

IEC 60601-1 uses the term 'applied part' to refer to the part of the medical device which comes into physical contact with the patient in order for the device to carry out its intended function.

Applied parts are classified as Type B, Type BF or Type CF according to the nature of the device and the type of contact. Each classification has differing requirements from the point of view of protection against electrical shock.

Type CF is the most stringent classification, being required for those applications where the applied part is in direct conductive contact with the heart or other applications as considered necessary.

Type BF is less stringent than CF, and is generally for devices that have conductive contact with the patient, or having medium or long term contact with the patient.

Type B is the least stringent classification, and is used for applied parts that are generally not conductive and can be immediately released from the patient.

Type B applied parts may be connected to earth, while Type BF and CF are 'floating' and must be separated from earth.

5.6 Updating the spryTrack firmware

The update of the spryTrack firmware is done through USB. A tool is provided with the Atracsys *SDK*.

+Never downgrade a spryTrack firmware without the authorization of Atracsys. Always check the compatibility matrix between the *SDK* release number and the spryTrack firmware release number provided by Atracsys before proceed to an update.

The procedure is the following:

1. Download the spryTrack firmware zip file from Atracsys website (referred to as [firmware.zip] field in next steps)

2. Power on the spryTrack by connecting it to the USB *Power Injector* or through *USB-C* directly to host PC.
3. Wait for the right front led to be static green or until the OS recognizes the device.
4. Identify the serial number of the device you want to update. ([SN] field in the next steps)
5. On Windows, open a command prompt from within the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") and run:

   ```
   stk_programmer64.exe -a f -i [firmware.zip] -s [SN]
   ```

6. On Linux, open a shell and, from within the installation path of the *SDK*, run:

   ```
   ./stk_programmer64 -a f -i [firmware.zip] -s [SN]
   ```

7. Wait for the completion of the update program. **Ensure with the program outputs on the console that no failure occurs during the update.**
8. Once the program successfully terminated, the device is automatically rebooted. The device is then ready for use.

If the update is interrupted, reboot the device, wait for 5 minutes and restart the procedure.

If you are not able to correctly update your firmware device, please contact Atracsys support. Do not attempt to use, in any way, a device on which a firmware update failed.

6 Software installation

This chapter presents how to install the *SDK* used to communicate with the device. The installation package is provided as an installation wizard for Windows and a compressed tarball for Unix/Linux, and contains:

- A Graphical User Interface (*GUI*) demo application;
- A cross-platform *GUI* application allowing to acquire data for assessment of the optical tracking device trueness (the *AVT* software);
- Headers and library files to integrate the driver in a project;
- Pre-compiled software examples, with the corresponding source code;
- Documentation of the Application Programming Interface (API);
- Some predefined *marker* files.

The latest installers are found on Atracsys website, using the username and password provided by Atracsys.

6.1 Windows installation

The installer for Windows is a wizard which allows an easy installation with possible customisation of the setup. Depending on the target environment, Visual Studio redistributables will be automatically installed as well.

6.2 Linux installation

The Linux package consists of a compressed tarball.

```
1 tar -xJvf fusionTrack_v_4_1_1-gcc-5.4_x64.tar.lzma
2 cd fusionTrack_v_4_1_1-gcc-5.4_x64
```

Listing 6.1: Example of setting up the SDK.

Setting the `ATRACSYS_SDK_HOME` environment variable to the root SDK directory, i.e. the `fusionTrack_v_x_y_z-gcc-5.4_x64` folder will allow the demo and precompiled softwares to find the geometry files located in the `data` 'installation' folder. Note that setting this variable is mandatory for the *AVT* to work.

Mono (open source implementation of the Common Language Infrastructure .NET) is required to run the *AVT*. Mono is used to send back the *AVT* data to Atracsys for analysis.
By default, most Linux distributions will not let regular users access USB devices. The software samples as well as the command line tool to update the firmware will only operate when launched from the root account.

In order to allow regular users the access to the spryTrack, a new rule has to be set for the udev deamon. On Debian system, one can run the commands listed in Listing 6.2 to create the proper udev rule.

```
1  sudo sh -c "echo 'SUBSYSTEM==\"usb\", ATTRS{idVendor}==\"1c82\", ATTRS{idProduct}==\"0200\
       ", MODE=\"0666\"' > /lib/udev/rules.d/51-atracsys-stk.rules"
2  sudo udevadm control --reload-rules
```

Listing 6.2: Commands to create a rule allowing non root users to access the spryTrack.

Please mind that to open the spryTrack demo software, the `QT4` libraries must be installed in your system.

```
1 libqtcore4
2 libqtgui4
3 libqt4-network
```

Listing 6.3: Packages mandatory to run the spryTrack demo.

These libraries can be found in `ppa:rock-core/qt4`.

7 How the spryTrack device works

This chapter describes the spryTrack device functioning.

7.1 Communicating with the spryTrack device

The spryTrack communicates with the host PC via USB 3.0 or with a wireless device via Bluetooth.

7.2 The spryTrack device coordinates system and working volume

The spryTrack uses a Cartesian right-handed coordinates system, described in Figure 7.1 and Figure 7.2. Its origin is the optical centre of the left camera. Unless specified otherwise, all distances are expressed in millimetres [mm]. An alternate coordinate system can be used, which center lies at the middle of the spryTrack device baseline, i.e. at the mid-point between the two cameras.

The spryTrack contains an accelerometer, which system of coordinates is the same as for the tracking data (since SDK version 4.7.1).

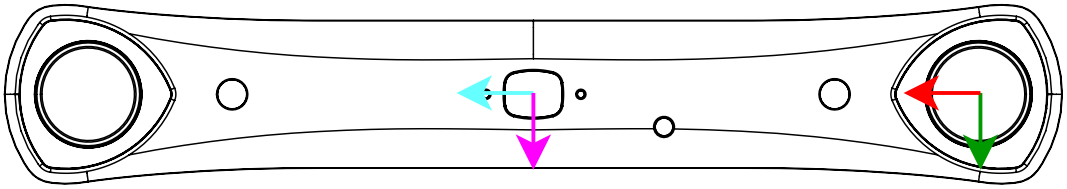


Figure 7.1: Coordinates systems, front view. The spryTrack device x axis is in red, the spryTrack device y axis is in green. The symmetrised x and y axes are in cyan and magenta respectively.

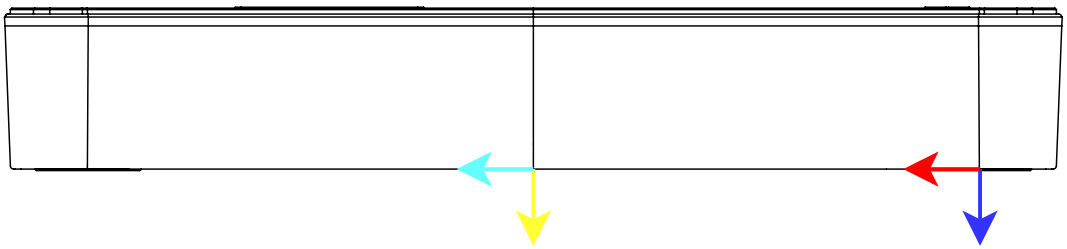


Figure 7.2: Coordinates systems, top view. The spryTrack device x axis is in red, the spryTrack device z axis is in blue. The symmetrised x and z axes are in cyan and yellow respectively.

In order to be tracked by the spryTrack device, the markers must lie inside the working volume described in Figure 7.3. The dimension of the working volume are shown in Figure 2.1. A 3D Step model is available on request.

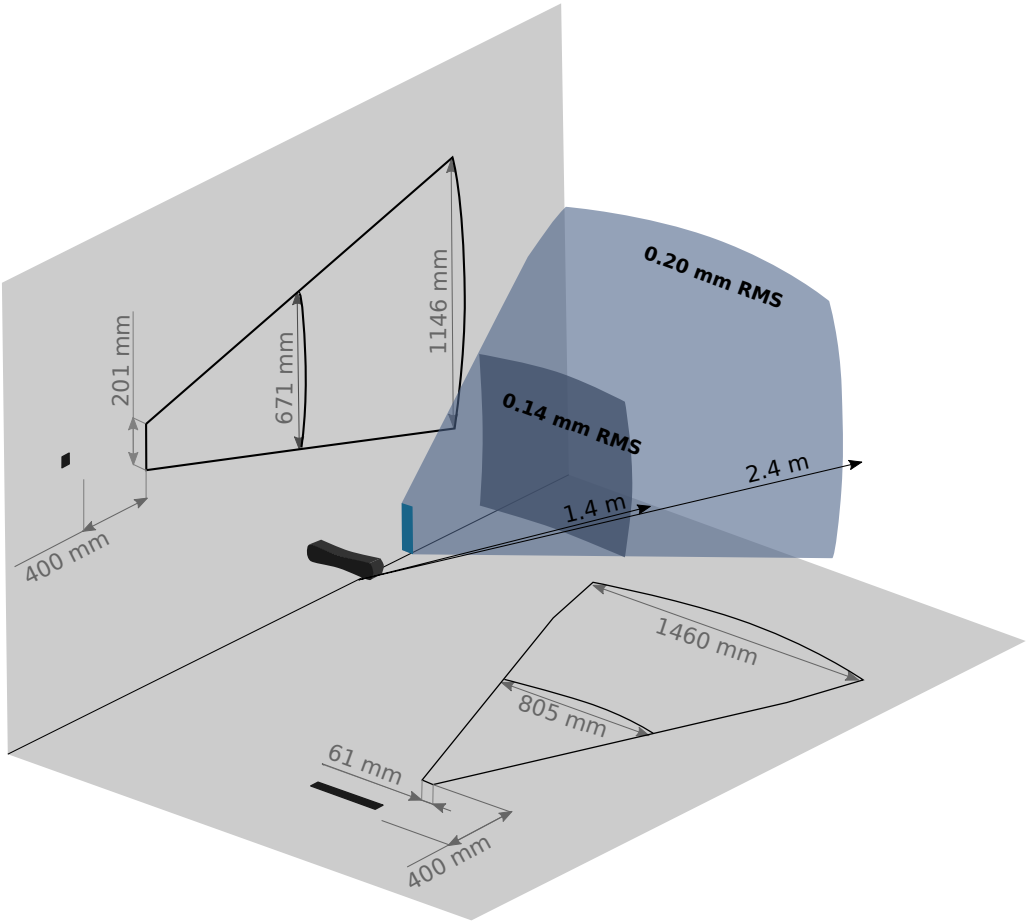


Figure 7.3: spryTrack 300 3D view of the measurement volume. The limits are described in Figure 2.1. The indicated truenesses are typical values measured on calibrated devices.

7.3 The spryTrack device lasers

To ease positioning objects in the working volume, the spryTrack device is equipped with an aiming system that integrates 2 visible laser pointers. Wherever the distance between the 2 laser dots is $\leq$ 15 mm, the centre of working volume is at a distance $\leq$ 105 mm, see Figure 7.4.

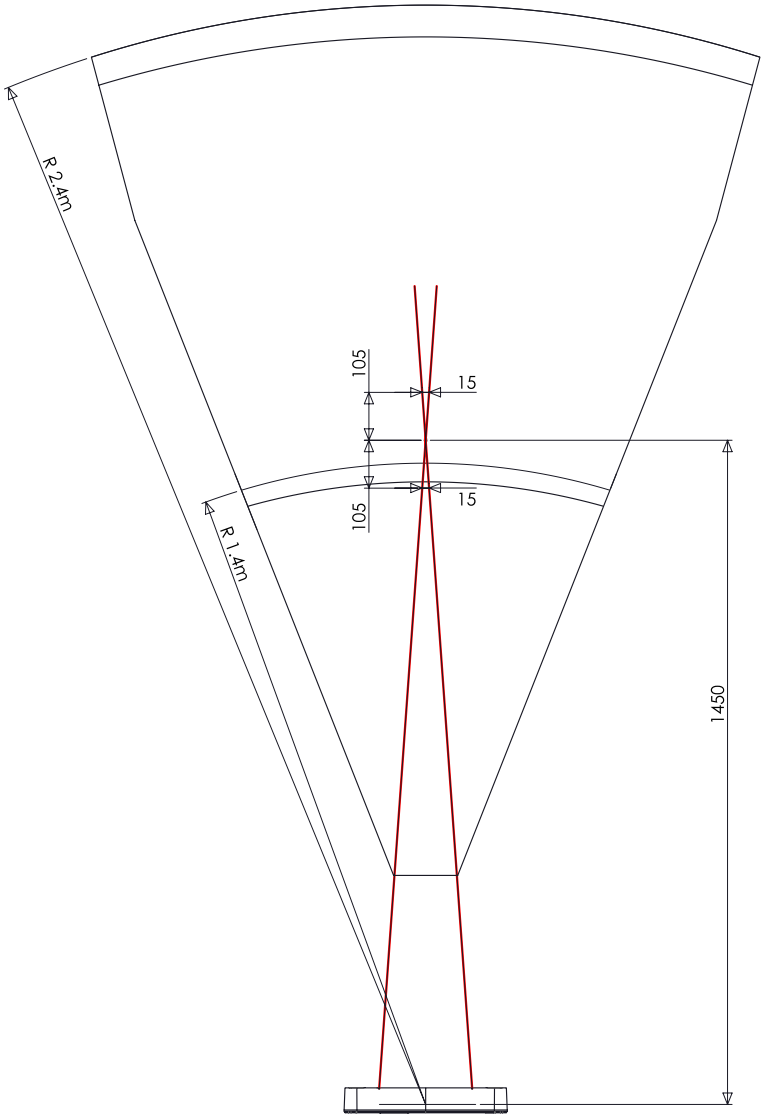


Figure 7.4: Laser crossing of spryTrack 300 to find center of working volume.

7.4 Information provided by the spryTrack device

The spryTrack is designed to provide accurate *relative* measurements, and should *not* be used to perform *absolute* measurement. See Chapter 3 for more information.

Most of the functions of the *API* return a status code, which indicates different levels of warnings and errors. The returned code *must* be checked by the Integrator's software implementation. The documentation is available in [1].

The various information retrieved from the spryTrack for each frame comes with a status flag, which must *always* be checked by the integrator's software implementation to ensure the integrity of the data.

The *API* is fully documented in [1]. In this section, the information which can be retrieved from the spryTrack when *markers* are tracked is shortly presented:

- The *marker* information, consisting of
  - the *marker* container status, indicating if an error occurred when retrieving the data;
  - the number of reconstructed *marker*, indicating the size of the container;
  - *marker* 3D position in mm;
  - *marker* orientation (3x3 rotation matrix);
  - registration error in mm, corresponding to the result of the least-square minimisation between the *marker* geometry and the measured *fiducial* positions;
  - the geometrical id, corresponding to the 'id' field from the geometry file Section 7.8;
  - a tracking id, giving a unique number to identify a *marker* on a frame;
  - a mask indicating which *fiducials* have been used to reconstruct the *marker*, i.e. if bit $\#i$ has value 1, it means that *fiducial* $\#i$ was present;
  - the indices of the used *fiducials*, allowing to get the information about the used *fiducial*;
  - the *status* of the *marker*, which indicates whether the marker lies in the accuracy measurement volume or not, consisting of the union of the statuses of all its *fiducials*;
- the *fiducial* information, consisting of
  - the *fiducial* container status, indicating if an error occurred when retrieving the data;
  - the number of reconstructed *fiducials*, indicating the size of the container;
  - *fiducial* 3D position in mm;
  - triangulation error in mm, corresponding to the error coming out of the triangulation algorithm, i.e. the error on the *fiducial* position;
  - epipolar error in pixels;
  - *fiducial* probability, indicating if a *raw data* was used to build multiple *fiducials*;
  - the indices of the left and right *raw data* used to reconstruct the *fiducial*, allowing to access the information about the used *raw data*;
  - the *status* of the fiducial, which indicates whether the *fiducial* lies in the accuracy measurement volume or not, unioned with the statuses of the two *raw datas* of which it consists;
- the left / right *raw data* information, consisting of
  - the *raw data* container status, indicating if an error occurred when retrieving the data;

  - the number of detected *raw data*, indicating the size of the container;
  - *raw data* 2D position in pixels;
  - *raw data* surface in pixels;
  - the *raw data* probability, based on the width / height ratio of the detected *raw data*;
  - the *raw data* status, indicating if the *fiducial* is located at an edge of the sensor, or if it lies outside the theoretical field of view;
- the picture header information, consisting of
  - the header status, indicating if an error occurred when retrieving the data;
  - the picture dimensions;
  - the picture counter, a unique and consecutive generated number;
  - the picture timestamp, corresponding to the time in µs since the device startup, the timestamp is taken at the start of the image exposure;
  - the picture image format;
- the left / right picture data, consisting of
  - the picture container status, indicating if an error occurred when retrieving the data;
  - the list of the picture pixels.

The position $\vec{x}$ and orientation R of the *marker* are interpreted in the following way (see Section A.2 for a full example), using the homogeneous notation (see Section A.1):

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix}$$

$$H = \begin{pmatrix} r_{00} & r_{01} & r_{02} & x \\ r_{10} & r_{11} & r_{12} & y \\ r_{20} & r_{21} & r_{22} & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where r_{ij} is the rotation matrix component at line i, row j.

The pictures are given as in PGM format, an example of using the data is shown on Listing 7.1.

```
err = ftkGetLastFrame( lib, serialNbr, frame );
if ( err == FTK_OK && frame->imageLeftStat == QS_OK )
{
  QImage* greyPicture( new QImage( frame->imageHeader.width, frame->imageHeader.height,
                                   QImage::Format_Indexed8 ) );
  for ( int row( 0 ); row < greyPicture->height(); ++row )
  {
    memcpy( greyPicture->scanLine( row ),
            frame->imageLeftPixels + row * frame->imageHeader.imageStrideInBytes,
            greyPicture->bytesPerLine() );
  }
}
```

Listing 7.1: Example of reading the picture using the Qt framework.

7.4.1 The spryTrack events

The firmware is able to send 'events' to the host PC. Those events have a type (defined in [1]), and optional data. Depending on the event type, they are periodically sent (the temperature-related events for instance), or sent on a specific condition on the device (when the temperature is too low or to high for example). The *SDK* provides a mechanism allowing the user to read those events.

Events generation and processing may have an impact on spryTrack latency. If too many events are forwarded in the same frame, this frame latency may not respect device specification. Please refer to section Section 8.4 for latency evaluation details.

7.5 Fiducial detection and marker tracking

Passive and active *fiducials* are detected by the spryTrack using different methods.
The spryTrack collects IR light for a period of time called integration time. This is an electronic shutter. The value of this integration time can be tuned via option.
For passive *markers*, during the whole integration time, IR illuminators are lit and the retro-reflecting coating of the passive *fiducials* reflects the IR light to the spryTrack sensors.
For active *fiducials*, the spryTrack emits an IR signal which triggers the LED emission on the connected *marker*.

In both cases, the IR light is detected by the spryTrack sensors in the same way. The spryTrack *SDK* then builds *fiducials* by combining pairs of *raw data*, wrong combinations are rejected using the epipolar error explained in Section 7.10. The reconstructed *fiducials* are finally combined into *markers*, using the provided geometry and applying the matching tolerance and a registration error cut to reject fake combinations.

The reconstructed *markers* are *tracked* from one frame to the next: this means a found *marker* on frame n will be looked for in frame $n + i$ around its last known position. This allows to save CPU cycles as the tracked *markers* do not undergo the whole reconstruction sequence.

The legacy *marker* reconstruction algorithm suffered from several flaws:

- a valid *marker* could be missed (i.e. not reconstructed) if, during the looking for *fiducial* i, an additional *fiducial* lay at about the same distance. The algorithm was able to see the *marker* could not be reconstructed due to the registration error but didn't look for another possibility;
- the 'Matching Tolerance' option (see Section 8.1.1) was not implemented correctly, resulting in the applied cut to be dependent on the compared distance in a non-trivial way. Due to the implementation, a simple fix was not possible;
- the *first* found candidate was chosen as the correct one. With large tolerances, this could lead to wronly reconstructed *markers*.

For those reasons, a new *marker* reconstruction algorithm was developped. Starting from SDK version 4.7.1, the new algorithm is used as default, the old one is still available for backward compatibility reasons. It can be chosen using a dedicated option: 'New marker reco algorithm'.

7.5.1 Active fiducials

As long as the wavelength is within the range specified (in Table 2.1) and the emitted power is sufficient, the device can work with several active fiducials (i.e. LEDs). In order to minimise angular dependency, only LEDs with maximum opening angle and without embedded lenses should be used. For non-medical / non-autoclavable use, Atracsys recommends the OSRAM SFH4250-Z LED, driven at 100 mA.

7.6 Temperature compensation

The internal temperature of the spryTrack device have a non-negligible rise time, which leads to a loss of trueness in both following situations:

1. as long as the device is not thermically stable;
2. if the steady temperature does not correspond to the temperature at which the calibration was performed.

In order to solve those issues, a temperature compensation algorithm is used. The compensation decreases the loss of trueness in the warm-up phase and if the reached temperature is different from the temperature during the calibration.

The integrator must check the 'Calibration type' option value corresponds to `CalibrationType::TemperatureCompensation` if and only if they want to use the temperature compensation.

The temperature compensation consists of multiple calibrations, each of them obtained at a different temperature. The spryTrack device has several temperature sensors, which values are combined to compute a 'synthetic temperature'. The 'temperature compensated calibration' is therefore stored as a set of calibration parameters indexed by a 'synthetic temperature' value. On each frame, the current 'synthetic temperature' value is computed, and a calibration is either interpolated or extrapolated from the ones contained in the calibration file. New events (see Section 7.4.1) have been added to allow the user to:

- get the current 'synthetic temperature' and the value of the 'synthetic temperature' when the device is stable in a room at 20 °C (called the reference temperature), only provided for user information[1];
- know if the current 'synthetic temperature' value is either below the lowest one or above the highest one: those events are `FtkEventType::fetLowTemp` and `FtkEventType::fetHighTemp` respectively.

The events are the only way to know if the SDK is interpolation or extrapolation mode, no specific status codes are issued. The presence of those events do not necessarily mean the data must be discarded: for instance, as soon as the synthetic temperature gets below the lowest temperature, the event is generated, not taking into account the temperature difference (i.e. the difference being 0.1 °C or 10.0 °C does not make any difference).

Depending on the spryTrack device model, the temperature compensation might not be available.

Please verify with Atracsys if your device model and firmware/software release integrates the temperature compensation feature.

[1] Since the external temperature modifies the synthetic temperature value at thermal equilibrium, the latter cannot be used by the integrator in their regulation software.

7.7 The spryTrack device markers

The *markers* used for surgery must be bio-compatible.

A *marker* shall be treated as an independent medical device. Use *markers* as specified by Atracsys. Other risks related to *markers* with reflecting *fiducials* may occur depending on the technology used by the manufacturer, these risks cannot be treated by Atracsys.

Any liquid on *markers* such as body fluids and water could lead to wrong measurements. The integrator has to specify in the end user manual instructions about not handling *markers* with dirty gloves, as well as instructions for cleaning / replacing *markers* and / or *fiducials* after getting in contact with liquids.

A *marker* in the camera view must have a specific, appropriately sized and distinguishable geometry. The integrator has to ensure that each used *marker* should be easily distinguishable by end users. A label on each *marker* has to be inserted. The integrator has to specify in the end user manual that operators have to choose *markers* with different geometries.

A *marker* is assumed to be calibrated and sterile. *Fiducials* are assumed to be correctly fixed on the *marker*. Instructions about fixing *fiducials* on *markers* must be clearly specified by the integrator in the end user manual.

Sterile *markers*, *fiducials* or battery packaging must be opened inside the Operating Room (OR). The integrator has to specify in the user manual to open sterile *markers*, *fiducials* or batteries packages only in the OR as well as the manner how to open it to guarantee sterility.

Sterile *markers*, *fiducials* or batteries in a damaged package cannot be used because it might have been deteriorated during transport which could lead to patient infection. Instructions about not using sterile *markers*, *fiducials* or batteries from a damaged package must be clearly specified by the integrator in the end user manual.

Sterile *markers*, *fiducials* or batteries in a damaged package cannot be used because it might have been deteriorated during transport which could lead to patient infection. The integrator has to provide sterile *markers*, *fiducials* or batteries with packaging presenting a certain robustness.

Markers or *fiducials* geometrically deformed due to shocks or manufacturing defects can create wrong measurements leading to patient injury. Instructions about not using geometrically deformed *markers* or *fiducials* must be clearly specified in the end user manual.

The integrator should define a zeroing method (see Chapter 11) or any equivalent method to verify marker integrity for all cases including corner cases of its application.

Markers or *fiducials* too close from lenses might lead to wrong measurement (see Table 2.1). The integrator has to mention in the end user manual that *markers* or *fiducials* should not be positioned too close from the lenses.

The integrator has to ensure generating an error message when the *markers* are placed on vibrating instruments and the frequency of vibrations could affect the measurements.

At all time, the integrator software should monitor the registration error of the reconstructed *markers*, to prevent badly reconstructed data to be used.

Instructions about using a calibrated *marker* and not geometrically deformed have to be clearly specified by the integrator to the user.

Integrator must implement necessary actions in order to avoid wrong reconstruction of *markers* (example: 2 fiducials of different *markers* mixed-up)

A *marker* is a rigid structure on which three or more *fiducials* are fixed so that the distance between them is constant.

The spryTrack system is compatible with both passive and active wireless *markers*. The spryTrack device identifies and tracks the *markers* using the geometry of the *fiducials* on the *marker*. The spryTrack device needs a geometrical description of each tracked *marker* (this description is called the geometry file).

Several causes related to *markers* might cause wrong measurements. Even when the *marker* itself has not been geometrically deformed, any damages on *fiducials* might be an issue: when a fiducial is particularly damaged, the *marker* might not be detected and tracked. In presence of a damage, the device is not able to match the *marker* with a known geometry.

A significant value of error is critical for an accurate tracking. In some cases, the *marker* might even disappear suddenly because the geometry is not identified anymore. In some cases, the *marker* in the field of view of the spryTrack could even not be detected. The integrator must indicate to the operator that the computed error is critical. It could be a warning or error message.

7.7.1 Passive markers

The passive *markers* use reflecting material in various shapes (spheres or discs). The used passive material has a specific coating which minimises the scattering of the IR light, allowing a large amount of it to be reflected back to its source. This causes the IR light from the illuminators to be reflected directly into the cameras. The spryTrack system is designed to track the position and orientation of *markers* and the position of individual reflecting objects.

For single-use *markers*, the end user manual must clearly state that the *markers* should not be reprocessed. The *markers* packaging must be marked with a label indicating they cannot be reprocessed.

For single-use *fiducials*, instructions about replacing new *fiducials* after every (single) use must be clearly specified by the integrator in the end user manual. Packages of passive *fiducials* have to be marked with a label indicating that they cannot be reprocessed.

A passive fiducial could be damaged due to a shock / scratch of the reflecting material. Instructions about not using deteriorated *fiducials* and how to replace a deteriorated passive *fiducial* must be clearly specified by the integrator in the end user manual.

Integrator has to specify in the end user manual instructions how to screw / plug *fiducials* on *markers*.

7.7.2 Active markers

The active *markers* consist of IR-LEDs mounted in a rigid structure. The LEDs can either emit continuously or be synchronised with the spryTrack cameras. In the latter case, the *marker* also contains an IR receiver. An active *marker* is powered by a battery, or from the equipment to which it is attached. Those *markers* can either be wired or wireless.

The spryTrack device IR communication sends a pattern which is recognised by the IR receiver of the active *marker*. Once the IR pulse is detected by the *marker*, the *marker* emits IR radiations, which is detected by the spryTrack device.

Active *marker* electronics has to be completely sealed in order to avoid liquid penetration due to cleaning. The integrator has to indicate in the end user manual instructions to clean active *markers*. The integrator has to specify in the end user manual that the operability verification procedure (see Chapter 11) must be done after each cleaning.

A geometrically deformed active *marker* due to a manufacturing defect could lead to wrong measurements causing patient injury. The integrator has to perform individual testing during *marker* calibration.

Active *markers* with batteries shall be handled with care. Operator who gets in contact with battery leakage incurs a risk of chemical burn. The integrator has to indicate in the end user manual that the battery of an active *marker* has to be removed after usage, especially in the case the active *marker* undergoes a sterilisation procedure.

The batteries used in active *markers* must comply with the regulation standards.

Battery for active *markers* must be sterile. The integrator has to indicate in the end user manual to use only sterilized batteries according to ISO 14937.

Any electric defects on active *markers* such as explosion, leakage, overheating might lead to operator or patient injury. The integrator has to use an adequate protection (e.g. fuse) that limits the current drawn from the battery. The integrator has to ensure that active *markers* contain a polarity marking and a polarity inversion circuit on the *marker*.

The measurement precision of an active *marker* can be detoriated because of liquids on its fiducials or IR-receptor. Instructions about identifying and cleaning of active fiducial must be clearly specified by the integrator in the end user manual.

An active fiducial *marker* must be sterilised before each usage. Cleaning or sterilisation methods instructions must be clearly specified by the integrator in the end user manual.

An active fiducial *marker* is able to transmit its individual parameters and data to the system. It can also notify the spryTrack with alert messages. An error message alerting that battery power is low is sent. That message must be handled by the integrator to alert the operator that the *marker* is not considered as operational due to a low battery and the *fiducials* must be turned off.

A weak optical signal from the IR-LEDs can lead to a wrong estimation of the position. The integrator should ensure the fiducials are turned off if the battery voltage goes under a given threshold.

Markers have to meet the IEC60601-1 requirement regarding 'Protection against excessive temperatures and other hazards'.

The integrator should use redundant *fiducials* (i.e. four or more *fiducials* instead of only three) and require the reconstructed *marker* to use all of them.

The battery must be removed from the *marker* after use, in order to prevent damage to the *marker* which may be caused by battery leakage.

7.8 Marker geometry files

In order to be able to track a new passive custom *marker*, a new *marker* geometry file must be registered in the SDK. This is performed using the `ftkSetRigidBody` function (or its legacy counterpart `ftkSetGeometry`), which documentation can be found in [1].

The demo application will automatically load the geometries from the installation directory and also from the 'user directory', as long as the file name is `geometryXXX.ini`. Active *markers* on the other side carries their geometries with them. The spryTrack retrieves the geometries directly after pairing.

The geometry of a *marker* consists at least of the position of each fiducial composing the *marker*. The *SDK* supports two file types (INI files [2], with basic support) and binary, and two different versions (versions 0 and 1), which sum up to three types of files:

1. legacy version 0 INI files;
2. INI files version 1, with additional stored data,
3. binary format version 1, equivalent to INI file version 1, used in the wireless markers, as they don't have much available memory.

The SDK allows to load the geometry information from those three file types, so that the user doesn't have to code the parsing.

The used coordinate system is a Cartesian right-handed coordinate system, which origin is reported as the *marker* position. All distances and positions are expressed in millimetres.

7.8.1 INI files version 0

The INI geometry file version 0 is the legacy format used by Atracsys since the begining of the spryTrack SDK. The file contains:

- a `[geometry]` section, containing:
  - a `count` key, which indicates the number of *fiducials* (at least 3, at most 6) composing the *marker*;
  - an `id` key, indicating the unique id of the geometry;
- A `[fiducialX]` section for each *fiducial*, where X goes from 0 to $n-1$, n being the number of *fiducials* composing the *marker* (described by the `count` key). Each `[fiducialX]` section contains the coordinates of the *fiducial* (in mm, no requirement on the origin) as follows:
  - an `x` key indicating the x coordinate of the position of the *fiducial*;
  - a `y` key indicating the y coordinate of the position of the *fiducial*;
  - a `z` key indicating the z coordinate of the position of the *fiducial*;

A user-defined file may contain comments (indicated by a '`;`' as first character) or additional sections, key-values pairs. The SDK will simply not take the additional data into account. In Listing 7.2 is given an example of a geometry version 0 INI file for a *marker* with 4 *fiducials*, which id is 998.

```
; Example of geometry file
[geometry]
count=4
id=998
[fiducial0]
x=-11.819
y=10.993
z=-0.006
[fiducial1]
x=-11.811
y=500.934
z=0.048
[fiducial2]
x=-11.77
y=970.879
z=0.266
[fiducial3]
x=-11.759
y=1165.952
z=-0.131
```

Listing 7.2: Example of geometry INI version 0 file.

7.8.2 INI files version 1

The INI geometry file version 1 extends the legacy format used by Atracsys since the begining of the spryTrack SDK, by allowing more information to be contained. The file contains:

- a `[geometry]` section, containing:
  - a `count` key, which indicates the number of *fiducials* (at least 3, at most 6) composing the *marker*;
  - an `id` key, indicating the unique id of the geometry;

  - a `version` key, containing the file version (i.e. `1`);
  - a `divotCount` key, indicating the number of divots (from 0 to 6) present on the marker;

- A `[fiducialX]` section for each *fiducial*, where X goes from 0 to $n-1$, n being the number of *fiducials* composing the *marker* (described by the `count` key). Each `[fiducialX]` section contains the coordinates of the *fiducial* (in mm, no requirement on the origin) as follows:
  - a mandatory `x` key indicating the x coordinate of the position of the *fiducial*;
  - a mandatory `y` key indicating the y coordinate of the position of the *fiducial*;
  - a mandatory `z` key indicating the z coordinate of the position of the *fiducial*;
  - an optional `normalX` key indicating the x component of the normal of the *fiducial*;
  - an optional `normalY` key indicating the y component of the normal of the *fiducial*;
  - an optional `normalZ` key indicating the z component of the normal of the *fiducial*;
  - an optional `angleOfView`, containing the half aperture angle of the cone in which the fiducial can be seen;
  - an optional `fiducialType` key, indicating the type of the fiducial (this corresponds to the underlying `uint32` value of the `ftkFiducialType enum` identifier;
- Optional `[divotX]` sections for each divot Figure 11.1, where X goes from 0 to $n-1$, n being the number of divots present on the *marker* (described by the `divotCount` key). Each `[divotX]` section contains the coordinates of the divot (in mm, no requirement on the origin) as follows:
  - a mandatory `x` key indicating the x coordinate of the position of the divot;
  - a mandatory `y` key indicating the y coordinate of the position of the divot;
  - a mandatory `z` key indicating the z coordinate of the position of the divot;

For each `[fiducialX]` section, either no `normal` key must be present or all `normalX`, `normalY` and `normalZ`.

In Listing 7.3 is given an example of a fake (the coordinates of the fiducials and divots actually make no sense) geometry version 1 INI file for a *marker* with 4 *fiducials* and 2 divots, which id is 42.

```
[geometry]
version=1
id=42
count=4
divotCount=2
[fiducial0]
x=0
y=11
z=3
normalX=0
normalY=1
normalZ=0
angleOfView=1
fiducialType=16
[fiducial1]
x=-33.72
y=-38.63
z=3
normalX=0
normalY=1
normalZ=0
```

```
angleOfView=1
fiducialType=16
[fiducial2]
x=0
y=-75.55
z=3
normalX=0
normalY=1
normalZ=0
angleOfView=1
fiducialType=16
[fiducial3]
x=41.28
y=-39.21
z=3
normalX=0
normalY=1
normalZ=0
angleOfView=1
fiducialType=16
[divot0]
x=0
y=-13
z=3
[divot1]
x=5
y=-1
z=3
```

Listing 7.3: Example of geometry INI version 1 file.

7.8.3 Binary files version 1

Binary files are not meant to be read by the user, they are only meant to be stored on devices with limited memory (i.e. wireless markers). The information in the file is exactly the same as in INI files version 1, and the conversion from binary (version 1) to INI (version 1) is invertible.

7.9 Marker tracking parameters

The tracking of *markers* (and *fiducials*) is tuned by a small set of parameters:

- epipolar max distance;
- matching tolerance / distance matching tolerance;
- registration mean error;
- matching maximum missing points;
- tracking range;
- tracking time span.

In this section those parameters are explained.

Epipolar max distance The epipolar max distance affects the *raw data* pairing. The epipolar error tolerance expresses the distance (in pixels) between the centroid of the *raw data* from the right camera and its expected position deduced from the centroid position on the left camera. The bigger the tolerance, the higher the probability of finding a matching pair, however the probability of creating a 'phantom fiducial' (see Section 7.10) is also increased. Finally, the number of performed operations increases with the epipolar error tolerance, as more possible combinations have to be tested.

Matching tolerance / Distance matching tolerance The matching tolerance affects how *fiducials* are matched to a known geometry. The matching tolerance is the maximal difference between the length between two *fiducials* and the reference length, as deduced from the geometry file. The bigger the tolerance, the looser the matching criteria. The 'Matching tolerance' option steers the behaviour or the old marker reconstruction matching algorithm, whilst the 'Distance matching tolerance' steers the behaviour of the new one.

Registration mean error The registration mean error sets a tolerance on the error returned by the *marker* registration algorithm. The computed error is a measurement of the deformation between the reconstructed *marker* and the theoretical geometry, linked to the accuracy of the positioning of the *marker*. The bigger the registration error, the looser the requirement on the system accuracy.

Matching maximum missing points The matching maximum missing points simply sets how many *fiducials* can be missed for a *marker* to be reconstructed by the system. When set to zero, every *marker* candidate for which a *fiducial* cannot be associated (due to the matching tolerance for instance) or is missing (e.g. occultation) won't be available in the frame data.

Tracking range The tracking range sets how far the *marker* tracking engine is allowed to look for a *raw data* when trying to find a *marker* from its last known position. This distance is measured in pixels, and must be adapted to the typical movement speed of the *marker* in real usage and the application frame rate.

Tracking time span The tracking time span sets over how many frames the tracking engine is allowed to look for a *raw data* when trying to find a *marker* from its last known position. As the *marker* are only reconstructed when a call to `ftkGetLastFrame` is performed, frames might be skipped if the application refresh rate is lower than the device acquisition frequency. The tracking time span can be seen as a tolerance on the number of skipped frame.

7.10 Phantom fiducials

A phantom fiducial is an artefact created by the triangulation algorithm, caused by pairing a wrong combination of *raw data*. This can happen when several *fiducials* are located on the same epipolar line. At the time of the triangulation, there is no way to know if a reconstructed fiducial is a real one (i.e. corresponding to a physical fiducial) or a phantom. *fiducials* sharing a raw data have a probability smaller than 1 (one): if a *raw data* is used n times, the probability of the resulting *fiducials* is set to $\frac{1}{n}$. During the registration stage, if any of the 'clones' is used to reconstruct a *marker*, its siblings are removed from the data.

7.11 Stray fiducials

A stray fiducial is a fiducial which was not used to reconstruct a *marker*. The *SDK* allows to get the 3D position of such *fiducials*. The stray *fiducials* may include phantom *fiducials* (see definition in Section 7.10), and the application must take care of removing them, and checking the validity of the reported positions.

Using stray fiducial positions may lead to hazardous situations, such as:

- IR transmitter, incandescent light, reflections or other IR source may be reconstructed as a stray fiducial;
- There is no way to identify a stray fiducial, i.e. its id (number) may change from one frame to the next. The application must therefore perform checks based on the position to ensure an identification of the fiducial;
- There is no way to distinguish a real stray fiducial from a phantom fiducial, only the probability defined in Section 7.10 is available.

8 Using the spryTrack system

In this chapter is explained how to use the spryTrack. It is assumed that the spryTrack is correctly setup as described in Chapter 5 and the software on the host PC is correctly installed as indicated in Chapter 6. The spryTrack is started by plugging it with a USB power cable. The spryTrack device is ready to be used once:

- the device beeps;
- the device status LED (see Section 3.2.3) appears in solid green or blue.

If the device status LED blinks or is in a different color, if the spryTrack beeps continuously or the situation is not corresponding to the described working state, please refer to Chapter 15. Switching off the device is simply done by unplugging the USB cable.

The default parameters have been chosen to ensure optimal tracking performances. Changing them might impact the performances of the spryTrack system.

The spryTrack device must be cleaned as described in Chapter 14.

To help the integrator to understand how to use the spryTrack and how to configure it, Atracsys provides:

- **Example sample softwares**: The samples are lightweight applications using the *SDK* libraries. They are provided as compiled applications and as source code. They showcase the main features of the *SDK*, and can be used as a base to create a custom application. They can be found in the relative path `spryTrack SDK/bin` . See the Section 8.3 for more details.
- **The ”demo” GUI**: The demo GUI displays in real-time the data from the spryTrack, and allows to configure the system graphically. It is an intuitive way to discover the features of the device and it may help in a debug process. The binary can be found in the relative path `spryTrack SDK/bin` . See the Section 8.2 for more details.
- **Options**: The option interface system is a feature of the *SDK*. The options are used to configure the *SDK* and the spryTrack device. They provide a unified interface for a broad range of use cases: they range from the configuration of the user LED in the device, to the configuration of the parameters of the triangulation algorithms in the *SDK*. See the Section 8.1 for more details.

8.1 The spryTrack system options

This section describes the options available for the spryTrack. It must be noted that some options are not available depending on the exact spryTrack model and firmware version. The options are split into five categories:

1. Library related options (e.g. Library version);
2. Device related options (e.g. integration time);
3. Detection stage related options (e.g. min / max size of the blobs);
4. 3D reconstruction related options (e.g. epipolar tolerance);
5. Wireless related options (e.g. marker button status streaming).

Each option has a type (`int32`, `float` or `ftkBuffer`), a read / write status and an optional 'global' flag indicating whether the option is global, in which case it is handled at the library level and must be accessed by specifying `0uLL` as serial number.

Unless explicitely flagged as 'permanent' in the option description, the set values are reset if the connection to the spryTrack device is closed (i.e. after a call to `ftkClose`).

Options names and IDs All options are identified by a unique ID of type `uint32`. The mapping between the option and its unique ID is not necessarily preserved between releases, hence the option enumerating mechanism (i.e. the `ftkEnumerateOptions` function) which allows to get the unique ID of each option at runtime (see also the 'ListOptions' software described in Section 8.3). Therefore, it is strongly recommanded to get the option list and the associated name-ID mapping for each option at runtime with `ftkEnumerateOptions`, and then, across the code when getting/setting an option, to call the options with their string names instead of their IDs.

Option configuration delay When setting an option value, the effect of the option may not be instantaneous, and there is especially no guarantee that the option will take effect in the next frame provided by the device.

When the application requires that an option is set, the integrator must not assume that the configuration is done when exiting the ftkSetInt32() function, but must ensure that the option is applied before using the data.

Note on the different types of spryTrack: There are 3 different types of spryTrack devices provided by Atracsys:

- The "IR" devices, which can only acquire images in the Infra-Red spectrum, and which are only aimed for Markers/Fiducials position acquisition.
- The "BW" devices (Black and White), which have an extended spectrum covering the visible spectrum, and which are made for Markers/Fiducials position acquisition and for standard image acquisition in the visible spectrum in black and white.
- The "RGB" devices (Red Green Blue), which also cover the visible spectrum with a color sensor to be able to send color images, and which also support Markers/Fiducials position acquisition.

The feature ”Dot projectors”, allows the user to project a determined light pattern on the scene to ease the frame analysis for Augmented/virtual Reality applications. Please note that the ”Dot projectors are compatible only with a spryTrack 300 and depending on the model, this feature might not be available.

These different types of devices are able to send different types of images:

- ”IR” frames, which can be sent by IR, BW, and RGB devices, which are aimed for Markers/Fiducials position acquisition.
- ”VIS” (VIsible Spectrum) frames, which can be sent by the BW and RGB devices, and which are aimed to be used as raw images (in black and white or in color depending on the model), and which are not processed for Markers/Fiducials position acquisition.
- ”SL” (Structured Light) frames, which can be sent by the BW and RGB devices, which have the same properties as the VIS frames. Additionnaly, the dot-pattern projectors (only forspryTrack 300) can be turned on (with settable strobe time by the user) during these frames.

In the BW and RGB devices, the order of acquisition of these types of frames over time can be configured by the user, see the option ”Image Scheduler Pattern” for more details.

8.1.1 Library related options

This subsection presents the library related options, which are options not directly linked to a specific device.

Name	Type	Description
Library Version	`ftkBuffer`, read-only, global	Allows to read the SDK full version string.
Device library version	`ftkBuffer`, read-only, global	Allows to read the device layer library full version string.
Data directory	`ftkBuffer`, read/write, global	Allows to access the path from where the marker calibration files and AVT software parameters files are retrieved.
Frame processing wall-time	`int32`, read / write	Allows to set a walltime on the frame processing. The default value (-1) set no walltime (which is the legacy behaviour). Any positive number n is interpreted as follows: the walltime will be $n \cdot$ 100 µs. If a walltime is set, the `ftkGetLastFrame` function can return the `ftkError_FTK_ERR_ALGORITHMIC_WALLTIME` if the set value is exceeded, and the gotten frame will contain no data but the picture header.

8.1.2 Device related core options

This subsection presents the device related options, which tune the behaviour of a specific device. Some of those are permanent, i.e. are only reset after a reboot of the device. The core options are common to IR, BW, and RGB devices.

Basic device related core options

Name	Type	Description
Device software version	`ftkBuffer`, read-only	Allows to read the general software version of the device, in string format.
Hard version	`ftkBuffer`, read-only	Allows to read the hardware version of the device, in a table of 4 uint8.
Shock monitoring version	`ftkBuffer`, read-only	Allows to read the software version of the shock sensor module of the device, in a table of 4 uint8.
USB module version	`ftkBuffer`, read-only	Allows to read the software version of the USB module of the device, in a table of 4 uint8.
BLE module version	`ftkBuffer`, read-only	Allows to read the software version of the BLE module of the device, in a table of 4 uint8.
Power delivery version	`ftkBuffer`, read-only	Allows to read the software version of the USB Power delivery module of the device, in a table of 4 uint8.
sTk device type	`ftkBuffer`, read-only	Allows to read the sTk model (180 or 300) and device capabilities (IR, BW or RGB) in a string
Factory file hash	`int32`, read-only	Allows to read the CRC32 hash of the content of the sTk factory file. This information may be request by Atracsys support team.
Image Compression Threshold	`int32`, read / write, permanent	Allows to access the picture compression threshold, i.e. minimum pixel intensity for a pixel to be considered white. Increasing the value decreases the compressed picture size.
Image Integration Time	`int32`, read / write, permanent	Allows to access the sensor integration time, i.e. the exposure time for the pictures. If the strobing is enabled, the strobe time of IR-LED is automatically synchronised.
BT MAC Address	`ftkBuffer`, read-only	Allows to read the MAC address of the BLE module, in a string format.
Enables IR strobe	`int32`, read / write, permanent	Allows to toggle on/off the IR strobe on each frame.
Enable embedded processing	`int32`, read / write, permanent	When this option is 1 (default), the frame analysis and marker triangulation is performed in the device, leading to less CPU load on the host PC, otherwise the triangulation is performed in the SDK.
Enable 16 bits pictures	`int32`, read / write	Allows to toggle on/off the 16 bits depth of the images sent back through the API. When set to OFF, the pixels in the images retrieved through the API are encoded on 8 bits. When set to ON, the same images are encoded on 16 bits (but only 12 bits are used). Changing this value does not affect the tracking.

Name	Type	Description
Enables lasers	`int32`, read / write, permanent	Allows to switch on/off the two pointing lasers of the spryTrack device. A value of 0 disables both lasers, 1 enables the left laser, 2 enables the right laser whilst 3 enables both of them.
Enable images sending	`int32`, read / write, permanent	When this field is 1 (default), the device sends the raw images, otherwise only the markers/fiducials data is sent.
User-LED red/-green/blue component	`int32`, read / write, permanent	Allow to access the red, green or blue component of the user LED. It has no effect as long as 'User-LED enable' is set to 0.[1]
User-LED frequency	`int32`, read / write, permanent	Allows to access the blinking 'frequency' of the user LED. A value of 0 disables the blinking, otherwise the higher the value the faster the LED blinks. It has no effect as long as 'User-LED enable' is set to 0.[2]
Enables the user-LED	`int32`, read / write, permanent	Allows to toggle on/off the user LED.[3]
Buzzer period	`int32`, read / write, permanent	Allow to access the device buzzer sound period (i.e. inverse of frequency) register. It has no effect as long as 'Buzzer repetition' is not set to at least 1.
Buzzer duration	`int32`, read / write, permanent	Allow to access the device buzzer sound duration register. It has no effect as long as 'Buzzer repetition' is not set to at least 1.
Buzzer repetition	`int32`, write	Start the buzzer with the given repetition. The buzzer period and duration are controlled with 'Buzzer period' and 'Buzzer duration' options.
Real time clock	`ftkBuffer`, read / write, permanent	Allows to read or set the internal real time clock (RTC). The RTC is part of the shock sensor module (see Chapter 4).
Shock sensor battery level	`float32`, read-only	Allows to read the last shock sensor circuit battery level [Volt] detected during device power off time (see Chapter 4).
SKU number	`int32`, read-only	Allows to read the spryTrack device SKU number.
BLE passkey	`ftkBuffer`, read /write , permanent	Allows to manage the spryTrack BLE passkey. The passkey change will be effective at next boot.
BLE NFC bonding	`int32`, read /write , permanent	Allows to limit the BLE bonding mode to 'NFC only' mode. Refers to section 8.6.8 for details.

[1] Only available for the spryTrack and fusionTrack version 00.10.00 and more recent.
[2] See note 1
[3] See note 1

Name	Type	Description
BLE images request	`int32`, write-only	Request the transfert of the last acquired scene image via a BLE connection. Write '1' to request the image of the left camera, '2' for the right one, and '3' for both. Sending Scene images via BLE could take a lot of time (up to 60s per camera) depending of the scene saturation.
Calibration type	`int32`, read-only	Allows to read the type of the calibration file stored in the device. The value corresponds to the `CalibrationType` enumeration.
Calibration processing datetime	`int32`, read-only	Allows to read the date and time of the calibration processing (given in UTC).
Calibration export	`int32`, read-write	Allows to set the export of the calibration parameters in the frame, so that they can be read using the `ftkExtractFrameInfo` function.
AR calibration info	`int32`, read / write, permanent	Camera pinhole model from camera calibration (see the `ftkARCalibrationParameters` structure in `ftkInterface.h`) for Augmented Reality applications.
Acquisition frequency	`int32`, read-only	Allows to read the spryTrack device acquisition frequency, in Hz.
Ignore hard working volume cuts	`int32`, read / write, permanent	Ignore rejection of fiducial data when they are not in the working volume (too close, too far, ...)
Save environment	`ftkBuffer`, write-only	This option allows to save the value of *all* read-/write options to a *JavaScript Object Notation (JSON)* file, which can then be loaded to restore all settings at once. The `ftkBuffer` contains the path to the *JSON* file to create (see Section 8.1.4).
Load environment	`ftkBuffer`, write-only	This option allows to restore the value of *all* read-/write options from a *JSON* file. The `ftkBuffer` contains the path to the *JSON* file to load (see Section 8.1.4).

Detection stage related core options

This section presents the options which are related to the 3D reconstruction of points and markers.
In the spryTrack, the reconstruction can either be processed in the SDK or in the device. The detection stage options are duplicated, and the options with an "Embedded" prefix denote the options to tune the reconstruction processed in the device. The two options are completely decoupled from each other, and setting an "Embedded" reconstruction option while the "embedded processing" option is off has no effect. All "Embedded" detection options are permanent.

Name	Type	Description
[Embedded] Blob Minimum Aspect Ratio	`float32`, read / write [permanent]	Defines the the minimal aspect ratio to consider a *raw data* during the raw detection phase. For example, an aspect ratio of 1 means that the bounding box of the *raw data* is a square. An aspect ratio of 0.5 defines a rectangle with one edge of length x and the other of length $2 \cdot x$.
[Embedded] Blob Minimum Surface	`int32`, read / write [permanent]	Defines the minimal surface in pixels to consider a *raw data* during the raw detection phase. This threshold allows to reject small polluting regions in the picture.
[Embedded] Blob Maximum Surface	`int32`, read / write [permanent]	Defines the maximal surface in pixels to consider a *raw data* during the raw detection phase. This threshold allows to reject large polluting regions in the picture
[Embedded] Symmetrise coordinates	`int32`, read / write [permanent]	Allows to toggle on/off the symmetrised coordinate system. When on, the origin of the coordinate system is computed as the point between the left and the right cameras.
[Embedded] Epipolar Maximum Distance	`float32`, read / write [permanent]	Defines the maximum distance (in pixels) between the right *raw data* and the right epipolar line defined by the left candidate during the match search.
[Embedded] Matching Tolerance	`float32`, read / write [permanent]	Defines the tolerance in mm between the measured and reference lengths defined by two *fiducials* of a *marker*, during the *marker* matching phase. The behaviour of this option is not corresponding to its expected behaviour, i.e a value of 5 mm does actually *not* correspond to a tolerance of 5 mm on the matched distance. If this option is changed when the new algorithm is used, a warning is issued.
[Embedded] Registration Mean Error	`float32`, read / write [permanent]	Defines the tolerance in mm after registration of the *marker* geometry with the measured one.
[Embedded] Matching Maximum Missing Points	`int32`, read / write [permanent]	Defines the maximum numbers of missing points when reconstructing a *marker* candidate valid during the *marker* matching phase (see the 'Matching Maximum Missing Points' paragraph of Section 7.9 for more details).
[Embedded] Tracking range	`float32`, read / write [permanent]	Defines the range (in pixels) where to look for a *raw data* associated to a *marker* according to its position in the previous frame(s) (See 'Tracking time span').
[Embedded] Tracking time span	`int32`, read / write [permanent]	Defines how many frames in the past the algorithm will inspect in order to identify a *raw data* associated to a *marker*.

Name	Type	Description
[Embedded] New marker reco algorithm	`int32`, read / write [permanent]	Toggles between the old marker reconstruction algorithm (0) and the new one (1).
[Embedded] Distance matching tolerance	`float32`, read / write [permanent]	Defines the tolerance in mm between the measured and reference lengths defined by two *fiducials* of a *marker*, during the *marker* matching phase, when using the *new* marker reconstruction algorithm. If this option is changed when the old algorithm is used, a warning is issued.
Embedded Frame processing walltime	`int32`, read / write, permanent	Allows to set a walltime on the frame processing. The default value (-1) set no walltime (which is the legacy behaviour). Any positive number n is interpreted as follows: the walltime will be $n \cdot 100$ µs. If an embedded walltime is set, no marker / fiducial / raw data is returned by the `ftkGetLastFrame` function when the set value is exceeded.

Wireless related core options

This section presents the options related to the wireless markers.

Name	Type	Description
Active Wireless Pairing Enable	`int32`, read-write	Toggles on / off the wireless marker search by the spryTrack device. This option is not compatible with the 'Marker button streaming' and 'Marker battery streaming' ones.
Active Wireless Markers reset ID	`int32`, write-only	Resets a wireless marker using its short ID.
Active Wireless button statuses streaming	`int32`, read / write	Toggles the periodic scanning of the wireless button status and the its streaming in the `ftkFrameQuery::events` member. If the 'Enable pairing' option is enabled, the streaming is disabled, whatever the value of this option.
Active Wireless battery state streaming	`int32`, read / write	Toggle the periodic scanning of the wireless battery level and the its streaming in the `ftkFrameQuery::events` member. If the 'Enable pairing' option is enabled, the streaming is disabled, whatever the value of this option.
Active Wireless Markers enable mask	`int32`, read / write	Set which wireless markers are fired on each frame. The options is a bitfield, meaning that if bit i is set to 1, marker with short ID i will be on. Due to the maximum number of simulataneously tracked wireless markers (which is 16), only the 16 LSB are used.
Active Wireless Markers info	`ftkBuffer`, read-only	Reads the information on paired markers, in CSV format. The file contains a header explaining the content. The field separator is a comma ','.

Name	Type	Description
Active Wireless Markers reset mask	`int32`, write-only	Allows to unpair a set of markers based on their ID, with a bit mask: if bit i is 1, then marker of ID i will be unpaired.
Upgrade marker firmware	`int32`, write-only	Upgrade the firmware of an active marker present in the scene, if the spryTrack detect that it may be updated.
Force Upgrade marker firmware	`ftkBuffer`, write-only	Force the upgrade (or downgrade) the firmware of an active marker present in the scene, with the firmware packaged in the spryTrack device.

8.1.3 Device-specific options

This section presents the options related to a specific type of spryTrack. The options presented here are only available for BW and RGB devices types if not specified.

Name	Type	Description
Image Scheduler Pattern	`int32`, read / write, permanent	Allows to set the image acquisition type pattern. It consists of a string of 54 ASCII-encoded values, each character representing a frame type: 'I' for IR frames, 'V' for VIS frames, 'S' for structured light frames. The frame pattern is repeated indefinitely over time during acquisition.
Image Integration Time for VIS frames	`int32`, read / write, permanent	Allows to access the sensor integration time (exposure time for the pictures) specific to VIS frames. Note that the strobe time for the LED is decoupled from this option, see the "IR LED Strobe Time for VIS frames" option.
Image Integration Time for SL frames	`int32`, read / write, permanent	Allows to access the sensor integration time (exposure time for the pictures) specific to Structured Light (SL) frames. Note that the strobe time for the LED is decoupled from this option, see the "Dot projectors Strobe Time for SL frames" option.
Enable dot projectors	`int32`, read / write, permanent	Enables n pairs of dot-pattern projectors for structured light. Note that the Dot-pattern projectors strobe time must also be adjusted, see the "Dot projectors Strobe Time for SL frames" option.
Dot projectors Strobe Time for SL frames	`int32`, read / write, permanent	Allows to access the dot-pattern projector strobe time for the Structured Light frames. Note that the "Enable dot projectors" option is responsible of enabling the projectors (default 0).
IR LED Strobe Time for VIS frames	`int32`, read / write, permanent	Allows to access the LED strobe time for the VIS frames. Note that the "Enables IR strobe" option must be set so that the LEDs can strobe.

8.1.4 Environment and options

In SDK version 4.3.1 was introduced the concept of *environment*, which represents a snapshot of the settings of a spryTrack device. The environment is defined by all read *and* write options, which can then either be saved to or loaded from a *JSON* file. The usage of the environment is triggered by the two options *Save environment* and *Load environment*, which are write-only options and take the path of the *JSON* file to read / write. In addition to the option values, the file also contains the device firmware version and the SDK version used for generation. This allows to ensure the saved data are correctly applied when loaded.

8.2 User interface software 'demo.exe'

The Atracsys *SDK* (documented in [1]) is shipped with a precompiled (*GUI*) program. This program is run by executing `demo64.exe`. This program consists of a main window, which is composed of several panels, and a secondary window showing a 3D representation of the scene (Windows only).

The GUI demo software is not part of the spryTrack software system. It is only meant to provide a showcase of what can be achieved using the SDK. The GUI demo software does not belong to the spryTrack product.

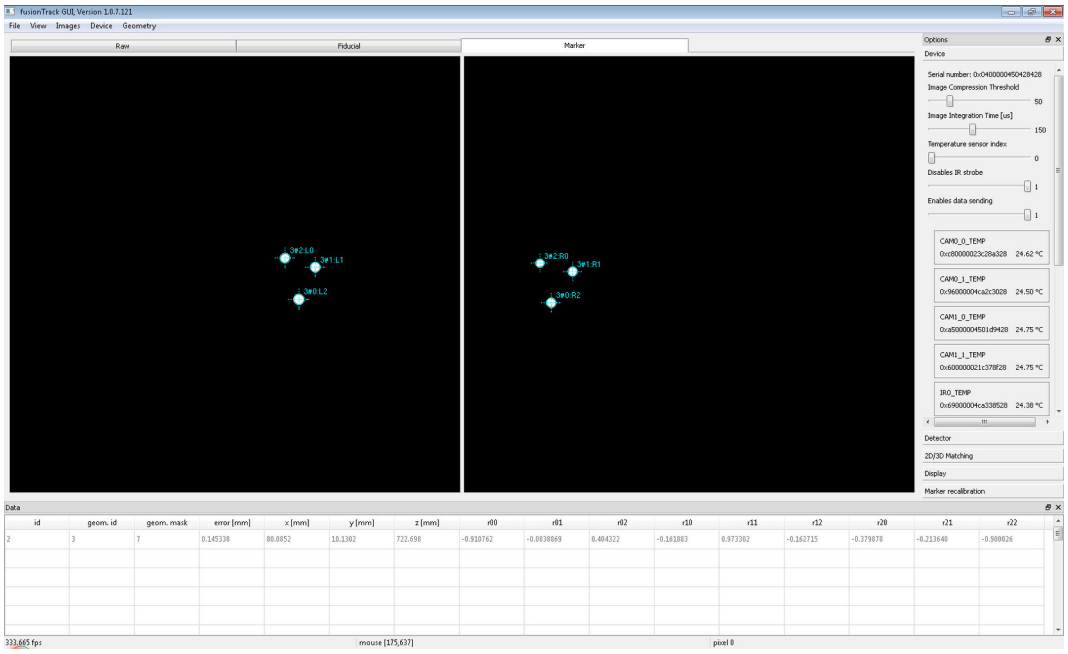


Figure 8.1: Running *GUI* program.

The menus allow to perform the following actions:

- 'File → Quit' closes the *GUI* program;
- 'View → Options' (re)opens the option panel;
- 'View → Data' (re)opens the data panel;
- 'View → 3D Viewer' (re)opens the 3D visualisation window;

- 'View → Log' (re)opens the log window;
- 'View → Reset Zoom' resets the zoom on the *two* pictures, this action can be achieved per picture by pressing the central mouse button over the desired picture;
- 'View → Save layout' saves the current layout (positions and sizes of the different panels and windows);
- 'View → Restore layout' restores the layout (position and sizes of the different panels and windows) from a previously saved state;
- 'Images → Save' allows to save pair(s) of pictures in the wanted directory, the picture file names are Left_X.png and Right_X.png, where X is an integer on 3 digits, automatically incremented;
- 'Geometry → Geometry X' (un)loads geometry with ID X. Unless specified differently during installation, geometry files are located at `C:\Program Files\Atracsys\spryTrack SDK\data` and user-defined geometries are loaded from `%APPDATA%\Atracsys\PassiveTrackingSDK` A greyed-out file indicates the file could be seen but not read (due to a syntax error for instance). When a geometry is loaded, it can be used with onboard or offboard processing. The geometry data is automatically sent to the spryTrack.
- 'About → Help' shows the GUI-specific help document.

Central panel The main window central panel shows the two pictures captured by the cameras. Just above the two pictures is located a tab selector, allowing to choose the type of information to show, corresponding to the data at different processing stages:

- *Raw data*, which are the blobs detected on each picture;
- 3D, which consist of a 3D point reconstructed from a pair of *raw data*;
- *Marker*, consisting of a 3D position and a rotation matrix (this is what is shown on Figure 8.1). The demo allows to open the two storage locations in order to see the files.

Lower panel Below the two pictures is located a dockable table, which shows data information. The shown data depends on the chosen type (i.e. Raw, *Fiducial* or *marker*). Part of the data (mainly position and indices) is replicated on the pictures, as superimposed information, whose colour depends on the data type: yellow is used for *raw data*, green for *fiducial* and cyan for *marker*.

Status bar The status bar is divided into five areas, which present (from left to right):

1. the device acquisition frequency in Hz;
2. the application refresh rate in Hz;
3. the coordinate of the cursor in the picture space[4];
4. the value of the hovered pixel[5];
5. the stability of the acquisition frequency and the device internal temperatures (when unstable the square is orange, blue is used for stable values).

[4]Only active if the ALT/meta key is pressed and the cursor is hovering the left or right picture.
[5]See note 4

Right panel On the right-hand side of the window is a dockable panel, allowing to change the values of the available options, perform *marker* recalibration, and export data.

The options are classed in four groups:

1. Device;
2. Wireless Active Markers.
3. Detector;
4. 2D/3D matching;
5. Display.

The Device-related option tab contains the following options:

- **Serial Number** which is the serial number of the tracking device. This option is read-only;
- **Image Compression Threshold**;
- **Image Integration Time**;
- **Enable IR strobe**;
- **Enable lasers**;
- **User LED red component**;
- **User LED green component**;
- **User LED blue component**;
- **User LED frequency**;
- **Enables the user LED**;
- **Shock monitoring period**;
- **Enable embedded processing** allows to enable or disable the computation of the markers and fiducials positions on the spryTrack;
- **Enable image sending** allows to enable or disable the sending of the images to the host. This is useful when on a USB2 connection and with the ”Enable embedded processing” option set to enable;

The Wireless Active Markers option tab contains the following options:

- **Paired markers** contains a table (See Figure 8.2) where paired *markers* are listed. The statuses of the buttons of the paired *markers* as well as the battery state are displayed. When a field is in RED, it means that the value has not been updated recently (2 second for the battery, 50ms for the buttons). The cause is most probably the *marker* being out of reach of the infrared transmitter and receptor of the spryTrack. Letting the mouse cursor on one of the battery field in the table will display a tooltip indicating the maximum and minimum voltages for the battery.
  The table also contains a column called ”Enable” with checkboxes. When checked, the spryTrack will activate the LEDs of a given marker at each cycle. This allows the tracking of the marker. When unchecked, the marker LEDs will never be lighted on. The tracking is therefore impossible. By default, the tracking is enabled after a pairing.
  When a row is selected in the table, additional information about the selected *markers* is displayed:

- **Serial**: The serial number of the *marker*. This number is usually also available on a sticker on the back of the *marker*.
- **Firmware Version**: The version of the firmware used in the *marker*.
- **Firmware Date**: The timestamp of the firmware used in the *marker*.
- **Model**: This value identifies the type of the *marker*. Currently three types are available: Wireless Boomerang (23), Wireless Pointer (73) and Wireless Development Kit (98).
- **PCB Version**: This value identifies the version of the electronic (the PCB) in the *marker*.
- **PCB Assembly**: This value identifies the version of the electronic parts assembled on the PCB.
- **Mechanical version**: This value identifies the version of the mechanicals parts of the *marker*.
- **Geometry**: The identifier of the geometry used for the tracking. The same value can be found in the data panel under the column ”geom.id”.

A button named ”Reset selected” under the table allows to unpair the selected *marker*.

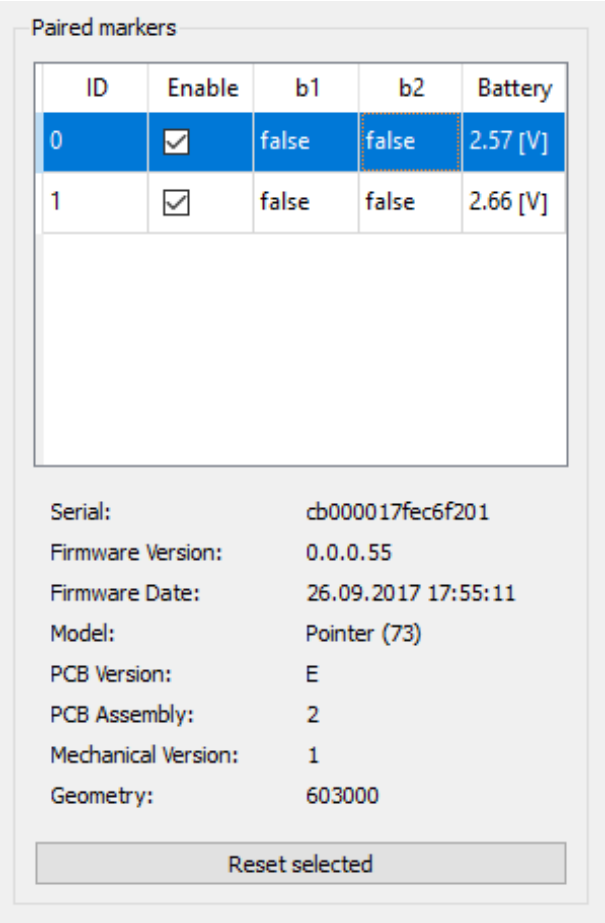


Figure 8.2: The Paired markers widget lists all the paired wireless markers and display their statuses.

- **Active Wireless Pairing Enable** allows to enable or disable the pairing of active *markers*. When enabled, any *marker* put in front of the spryTrack will be paired. While enabled, the performances are slightly degraded. Therefore, it should be disabled once all *markers* are paired.

- **Active Wireless button statuses streaming** allows to enable or disable the streaming of button statuses of each paired *markers*. When enabled, the spryTrack will ask every *markers* the statuses of their buttons (pressed or not) and stream it to the application at every frame.
- **Active Wireless battery state streaming** allows to enable or disable the streaming of the battery state of each paired *markers*. When enabled, the spryTrack will ask every *markers* the statuses of their battery and stream it to the application every second.

The next options (detector related and 2D/3D matching) are dedicated to the triangulation algorithm. This algorithm can run either on the spryTrack (when **Enable embedded processing** is ON) or on the host PC (when **Enable embedded processing** is OFF). For each options related to this algorithm, two values are displayed (with onboard and offboard labels).

The Detector-related option tab contains the following options:

- **Blob Minimum Aspect Ratio**;
- **Blob Minimum Surface**;
- **Blob Maximum Surface**.

The 2D/3D matching-related option tab contains the following options:

- **Epipolar Maximum Distance**;
- **Matching Tolerance**;
- **Registration Mean Error**;
- **Matching Maximum Missing Points**;
- **Tracking range**;
- **Tracking time span**.

The Display-related option tab gathers options which are specific to the *GUI*, i.e. they do not correspond to any option from the *SDK*. They are:

- **Display images** enables picture display. This can have an impact on the *GUI* framerate on low-end PCs;
- **Display error info in images** enables showing additional information on the picture, i.e.:
  - aspect ratio for the *raw data*;
  - epipolar and triangulation errors for *fiducials*;
  - registration error for *markers*;
- **Update Timer Timeout** allows to limit the *GUI* framerate, the value of the minimal interval between two calls to ftkGetLastFrame in ms.

8.2.1 Firmware upgrade of an Active Marker

The spryTrack firmware contains a copy of the latest active marker firmwares (for the dev kit, the Boomerang and the Pointer). When a paired active *marker* firmware is not up-to-date with the version embedded in the spryTrack, the demo application lets the user upgrade it. A button "Upgrade Selected" will be displayed below the paired devices table in the Wireless Active Markers option tab. When clicking on this button and starting the procedure, it is very important to follow carefully the instructions. Before upgrading the firmware, make sure that:

- The active *marker* to upgrade is in front of the tracker.
- There are no other active *markers* around.

Before clicking on the OK button to upgrade the firmware, the active *marker* status LED has to be in purple (meaning that the marker is booting). In order to set the *marker* in this booting state, simply pull out and in the battery.

When the firmware upgrade is complete (after approximately 20 seconds), the active *marker* will reboot.

If the active *marker* does not reboot (the status LED stays in purple) , it can to be recovered using the sample stk17_-ForceWirelessMarkerUpdate.

8.2.2 Recalibration of a marker

In order to get even more accurate measurements, the *markers* can be calibrated each time the reflecting *fiducials* are mounted / dismounted. The demo application allows the user to perform this operation. This operation requires that only the *marker* to recalibrate is seen by the spryTrack (and the corresponding geometry must be activated), it also requires the temperature of the device to be stable, i.e. the device must be running for some time. The procedure records the *fiducial* positions for a maximum of 10,000 frames and creates a new geometry file, which is saved automatically and can be opened directly from the demo application. It also will be automatically added to the list of known geometries and read at the next start of the demo application. The old geometry is automatically disabled and the new one enabled. The produced file is stored in a user-specific location (`%APPDATA%\Atracsys\PassiveTrackingSDK` on Windows).

The recalibration algorithm can reassign fiducial IDs, i.e. fiducial 0 from the old geometry might become fiducial 2 in the new one. This is done such that similar geometries have similar geometry files. The choice is based on distances to choose fiducials 0, 1 and 2. The first three fiducials are always coplanar.

8.2.3 Exporting the data

The demo program allows to export the received data in CSV format[6]. The dedicated panel 'Data export' allows to chose the exported data and the number of saved frames. The export consists of up to three files, respectively containing:

1. the *raw data*;
2. the 3D *fiducial* data;

[6]A comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. Plain text means that the file is interpreted as a sequence of characters, so that it is human-readable with a standard text editor. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

3. the *marker* data;
4. the temperature data.

It must be noted that the 'demo.exe' software acquires data from the spryTrack at the display update rate. This means that exported values won't be recorded at the maximum speed of the spryTrack. The user can choose which data to export (the default setting is to export everything, i.e. to create three files). The number of frames to record can be set, in which case the saving will automatically stop when this number is reached, or let free. In the latter case, the user will need to stop the recording manually. In order to ensure compatibility, a version of the export can be set, which garantees that a reading application will be able to read the data created by a newer version of the application. The separator used in the CSV files is the semi-colon (';'), to comply with systems using a comma as decimal mark. The saved information is the same as contained in the `ftkRawData`, `ftk3DFiducial` and `ftkMarker` structures. The written information for *raw data* is the timestamp (in hexadecimal), 'left' or 'right' as an indication of the camera, the index of the *raw data*, the 2D coordinates of the *raw data*, its surface, probability and status, as presented in Listing 8.1.

```
1 timestamp;left/right;index;x;y;surface;probability;status
```

Listing 8.1: Structure of a line of the raw data export CSV file.

The written information for *fiducials* is the timestamp (in hexadecimal), the index of the *fiducial*, its 3D position, the values of the epipolar and triangulation errors, its probability and the indices of the used left and right *raw data*s, as presented on Listing 8.2.

```
1 timestamp;index;x;y;z;epipolar_err;triangulation_err;probability;left_index;right_index
```

Listing 8.2: Structure of a line of the 3D *fiducial* export CSV file.

Finally, the written information for *markers*, shown in Listing 8.3, is the timestamp, followed by the index, the tracking id, the 3D position, the rotation (row-wise), the registration error, the geometry id, the *fiducial* presence mask and the corresponding *fiducial* index.

```
1 timestamp;index;tracking_id;x;y;z;rot[1-9];registration_err;geometry_id;presence_mask;
    fiducials[1-4]
```

Listing 8.3: Structure of a line of the *marker* data export CSV file.

The synchronisation of the data between the various files is achieved by a manual matching of the frame timestamp, present in all files.

8.2.4 Dumping the data

The demo software allows to dump all data of a frame, for diagnostics purpose. The used file format is XML version 1.0, and the produced files contain all the needed information to reprocess the data. The user can enter:

- the number of frames to record (the value -1 is equivalent to inf, i.e. the user must stop the record);
- a file name prefix;
- a 'period' for picture saving, i.e. a picture pair will be saved every n frames (a value of 0 results in *no* saved pictures).

The 'Record' button triggers the choice of the output directory. Then the record starts until the user hits 'Stop recording' or the number of frames to record is reached. The file name consists of the prefix, followed by by the device serial number (in hexadecimal) and the current timestamp, e.g. `Dumper_0xbb000005c21ffa28_2017-11-07_14h39m53.xml`.

8.3 Command line software samples

The Atracsys *SDK* is also shipped with precompiled command line softwares, which show C++ code examples of how to use the *SDK*. From the install directory they can be found in the relative path `spryTrack SDK/bin` . Each sample demonstrates one operation. In order to run them, the spryTrack must first be switched on. Once the spryTrack is properly running, any of the command line samples can be run . The documentation of the various samples is present in [1], a short description is given here below.

Each software can take at least an additional configuration file as argument, using the `-c/--config path_to_file`. The configuration files are described in Section 8.9. This configuration file allows to set option values. Other arguments are described when calling the software with the `-h/--help` switch.

stk1_ListOptions exposes how to open/close the driver, enumerates the connected devices and lists the options for a device.

stk2_AcquisitionBasic shows how to load a geometry from a file and how to get the *marker*'s position and orientation. This program only produces output if the geometry of the presented *marker* matches the one in the file given in arguments, and shows the position and error for any found *marker* of that kind (units are mm).

stk3_AcquisitionAdvanced explains how to get the *marker*'s position and orientation, the *fiducials* positions and the raw data. Again this program requires the use of a *marker* and that its geometry corresponds to the geometry file given in argument. For each found *marker*, the error is printed and the position of the *fiducials* of which it consists.

stk4_AcquisitionRawData presents how to display raw data from the frame. This program does not require any *marker*, single or multiple *fiducials* can be used. The raw data for both cameras are printed, then the reconstructed *fiducials* are printed. As the program also tries to save a pair of pictures, it may require admin privileges to be fully functional on Windows.

stk5_AcquisitionRelative shows how to use one *marker* as reference for the transformation of a second one. This is the usual way to use the data in a *navigation system*. The two original transformations are printed, and the relative position and rotation of the 'slave' *marker* with respect to the 'master' one is computed.

stk6_ControlLED presents how to control the user LED, by setting the LED colour (the red, green and blue components can be set individually) and its blinking frequency.

stk7_AcquisitionExtended demonstrates how to get additional information on the current frame. The sample contain an example of how to get the real number of reconstructed 3D *fiducials*, even if no storage has been reserved for the 3D *fiducials*.

stk13_TemperatureMonitor displays the temperatures of the 8 sensors in the spryTrack. The temperature sensors are identified by a number, and Atracsys guarantees that a sensor with a given ID will always be located at the same place, and that there always will be *at least* one sensor in each of the following regions:

- the left part of the device;

- the right part of the device;
- the front part of the device;
- the bottom part of the device.

But there is no guarantee that a given ID (e.g. 1) will persist during the whole lifetime of the product (i.e. on all past, current or future versions of the product). The current positions for the sensors are illustrated on Figure 8.3, Figure 8.4 and Figure 8.5.

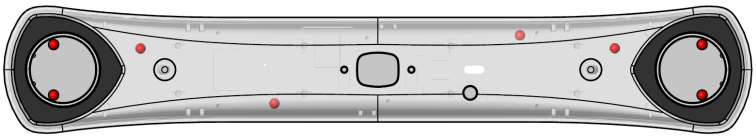


Figure 8.3: Temperature sensors location, front view.

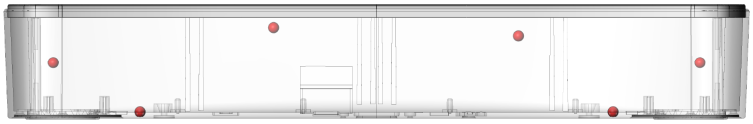


Figure 8.4: Temperature sensors location, top view.

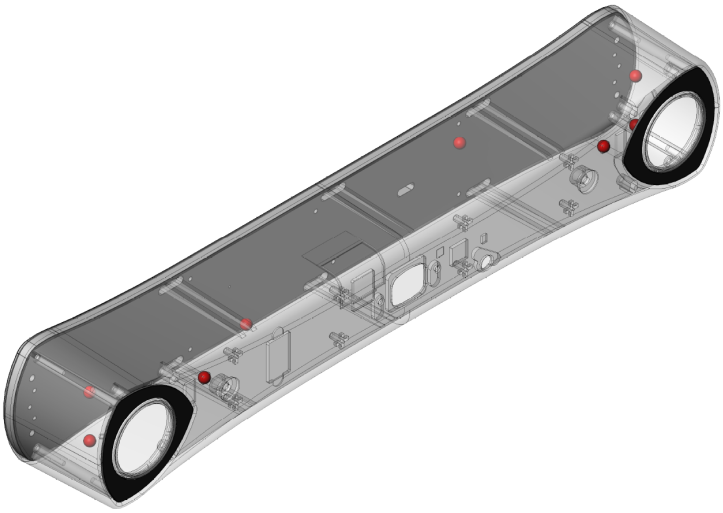


Figure 8.5: Temperature sensors location, isometric view.

stk15_GetAcceleration presents how to get the current acceleration values from the accelerometers.

stk16_InitializeRealTimeClock syncs the local time of the PC on which the sample is ran with the real time clock of the spryTrack. It needs to be run only once. Once set, the timings in the logs retrieved through the stk12_TroubleShooting sample will be correct. The shock events retrieved through the sample stk15_GetShockMonitoringData will also have correct timings.

stk17_ForceWirelessMarkerUpdate forces the upgrade of an active wireless *marker* with the firmware version packaged within the spryTrack. This sample takes as argument the model of the active *marker* (wTp for pointer , wBo for Boomerang and wTm for dev kits). Before running this sample, reboot the active *marker* by pulling out and in the battery. When the status led of the active *marker* is purple, the sample can be launched. The active *marker* will reboot once the firmware is updated.

stk18_DumpDeviceInfo shows how to get a dump of the device information. In case of a problem, this dump can be sent to Atracsys for diagnostics. The software creates an XML file, which contains the following information (non-exhaustive list):

- date and time;
- device serial number and type;
- SDK version;
- values of the various options;

stk19_OpticalCommunication presents how the to get a wireless marker battery and button status, using events.

stk22_EnvironmentHandling shows how to save and load environment using the C++ API.

stk23_Get16BitsImage saves one image from the left sensor and one image from the right sensor with a depth of 16 bits. On the 16 bits, only 12 are used. Changing this option does not affect the tracking. The demo software saves images with a depth of 8 bits to maintain the compatibility with the fusionTrack line of devices.

stk25_ImageScheduler sets the image type scheduler so that the spryTrack alternates between sending standard IR frames, Structured Light frames, and Visible Spectrum frames.

stk28_BluetoothSecurityOnConnexionManagement provides an example on how to handle the spryTrack BLE passkey and *Near-Field-Communication* mode. Please refers to 8.6.2 for details.

stk29_ExtractCalibrationParameters demonstrates how to perform the extraction of the calibration parameters from the gotten `ftkFrameQuery` instance. Please note that the distortion parameters order does follow the Matlab calibration toolbox convention, which differs from the openCV one.

stk30_Latency demonstrate how to evaluate the latency of the spryTrack. Please refers to Section 8.4 for details.

stk31_UsbReconnection provides an example on how to handle the USB disconnection and reconnection during an acquisition loop.

stk32_InterpolateFrames allows to get an interpolated frame intercalated between the two input ones. The current implementation only interpolated the `ftkRawData`, `ftk3DFiducial` and `ftkMarker` instances corresponding to the `ftkMarker` instances found in *both* input frames, all other data will simply be discarded from the interpolated frame.

stk33_Buzzer provides an example on how to use the spryTrack internal buzzer by playing an ascending major scale.

stk34_ReprocessFrame demonstrates how the user can plug their own reprocessing code and get the SDK to recompute 3D fiducials and markers from the altered data.

8.4 Latency measuring

The spryTrack latency is defined as the time delay between the end of a frame exposure (during acquisition) and its reception (the return of ftkGetLastFrame()) on a host PC running the SDK. Several factors can impact the latency of the device, among which certain play a key role:

- The content of the scene (high number of markers, parasitic reflections…) that will directly impact the processing and transmission times.
- The embedded processing option being enabled or not.
- The performances of the host PC running the SDK.

To try estimating the latency, Atracsys provides an example which allows the integrator to start a measure of the device latency in its current environment and running platform: `stk30_Latency`, available both as a pre-compiled binary and source code.

The example follows this principle: while streaming, an environment with passive marker(s), the spryTrack will first disable its IR strobes and the software will ensure that no markers are detected in the frame data. Then it computes the time between the enabling of the IR strobes and the moment the marker(s) is correctly present in the frame data.

As per how the measure is performed, it actually measures the time difference between before starting the IR strobes and after receiving the actual data. In addition to the device latency, the time for the IR strobes to be turned on is adding a delay ranging from 0 and 18.5 ms: the enabling command will be issued during the acquisition of a frame and will be considered for the next frame acquisition. Depending on the moment the command will be received and interpreted by the spryTrack, at worst it would add 18.5 ms and at best 0 ms (18.5 ms being the acquisition time of a frame for the spryTrack).

To estimate at best the latency, the software needs to perform an important number of measures while inducing, between each measure, a random delay. This delay is added so that we are capable of sending the enable command of the IR strobes at all possible timing. This way, by the mean of the important number of measures performed, it is estimated that the upper latency found by the software corresponds to the sum of the upper-bound of the spryTrack worst latency in its current environment and running platform, and the worst delay added by the IR strobe enabling (18.5 ms). The software thus substracts 18.5ms from the worst measure found to provide the results.

Please note that the true non-biased device latency (without the addition of the delay to turn on the IR strobe) cannot be measured by this mean. The non-biased latency is only guaranteed to be lower than the sum of the upper bound of the latency provided by the software and 18.5ms. This software assumes that the biased and non-biased values will converge when the number of measures increase.

8.4.1 Running the software

The user can steer the behaviour of the software via the following options:

- `-g/- -geometry`: Geometry file(s) of the passive marker(s) to load.

- `-n/- -nbloops`: The number of measures to be performed by the software. At each measure, nbframes are captured.

- `-f/- -nbframes`: Number of frames to be recorded for a measure.

- `-l/- -maxLatencyTime`: The maximum acceptable latency for a frame in milliseconds.

This software is meant to be run in the user environment of use of the device to measure an estimation of the worst latency. It is recommended to a run the software for at least 1000 nbframes and 10 nbloops, which are the default parameters of the sample. An example of the sample is given in Listing 8.4.

```
./stk30_Latency64.exe -n 1 -f 100 -l 35 -g "C:\Program Files\Atracsys\spryTrack SDK x64\data
    \geometry002.ini"
================================================================================
                                    stk30_Latency
================================================================================

This software is the property of Atracsys LLC. Copyright (c) 2021 by Atracsys LLC.
This is a project to compute an upper-bound of the spryTrack frame latency.
For more information on the latency measure, please refer to the spryTrack user manual.

The processing is about to start. It should take around: 501 s.
Processing.....................................................................

Creating output file: ..\data\stk30_Latency.csv.
Output file created!

Test result:
-----------
The maximum acceptable latency defined is: 35 ms.
The minimum latency measured is: 23.6378 ms.
The maximum latency measured is: 29.7736 ms.
The maximum acceptable latency defined is respected.
```

Listing 8.4: Example of output.

In addition, the sample generates a CSV file in the data folder: `stk30_Latency` holding all the latency measured. The user can use this file to do an analysis of the maximum latency measured and at which frequency they occur. This can be helpful, for instance, to estimate if the number of measures performed are enough to validate the estimation of the worst latency.

8.5 Compilation of provided samples

The command-line samples discussed in Section 8.3 are shipped with their source code, in order to present how to use the *SDK* functions. The makefile are built using CMake. The cmake configuration file to create the makefile is located in the `fusionTrack SDK/samples` directory, lying in the installation folder. On windows, creating the makefiles in the default installation directory may require administrator rights.

Compiling the samples requires CMake version 3.14 or newer, and a C++11 compliant compiler (e.g. gcc 5.4 or newer, clang 3.8 or newer, Microsoft Visual Studio 2015 or newer).

The SDK comes with helper cmake files, which allows to simply use either the C or the C++ sdk with minimal efforts in a custom-made CMakeListst.txt file, as demonstrated in Listing 8.5.

```
/set(Atracsys_DIR "path_to_your_install")
find_package(Atracsys REQUIRED COMPONENTS SDK AdvancedAPI)

```

```
# Using the C API
add_executable(test_sample_c test_sample.cpp)
target_link_libraries(test_sample_c Atracsys::SDK)

if (NOT (MSVC AND MSVC_VERSION LESS 1900))
  # Using the C++ API
  add_executable(test_sample_c++ test_sample_advanced.cpp)
  target_link_libraries(test_sample_c++ Atracsys::AdvancedAPI)
endif ()
```

Listing 8.5: Example of linking against the C and C++ APIs.

The CMakeLists.txt file will be looking for the shared libraries in the SDK installation folder. In order to deploy an application built using the SDK, the device64 and fusionTrack64 shared libraries must be distributed as well. For the spryTrack, the libusb-1.0 shared library also needs to be deployed. This shared library is a custom build of libusb based on a pull request Atracsys made to support isochronous transfers on Windows (https://github.com/libusb/libusb/pull/284).

8.5.1 Windows compilation

The cmake command from the cygwin environment won't work, as it will use Unix-like pathes, which are not understandable by Visual Studio.

The procedure using Visual Studio is the following:

1. Create an empty build directory (in which file can be written) and open a terminal in this directory (either `cmd` or `PowerShell`);
2. From the aforementioned terminal, run `cmake -G <generator_string> [-A x64] <path_to_the_sample_dire` > command. Valid examples are

```
cmake -G "Visual Studio 15 2017 Win64" "C:\Program Files\Atracsys\fusionTrack SDK\
    samples"
cmake -G "Visual Studio 16 2019" -A x64 "C:\Program Files\Atracsys\fusionTrack SDK\
    samples"
```

3. Open the `samples.sln` solution and compile.

8.5.2 Unices compilation

A build system (e.g. GNU make, Ninja) is also needed on top of cmake and a compatible compiler.

The procedure is the following:

1. Untar the SDK tarball:

```
% tar xJvf fusionTrack_SDK-[...].tar.xz
```

2. Create a build directory and enter it:

```
% mkdir build
% cd build
```

3. Run cmake and compile, using the GNU Make build system:

```
% cmake -G "Unix Makefiles" -DCMAKE_BUILD_TYPE=Release ../fusionTrack_SDK-[...]/samples
%make -j <number_of_cpu>
```

or with the Ninja build system:

```
% cmake -G "Ninja" -DCMAKE_BUILD_TYPE=Release ../fusionTrack_SDK-[...]/samples
% ninja
```

8.6 Bluetooth Low Energy

When no host computer is available, the spryTrack can also operates without wires and without USB using a Bluetooth Low Energy connection. The Bluetooth Low Energy (BLE) protocol is defined by the Bluetooth Special Interest Group:

> Bluetooth® Low Energy (BLE) enables short-burst wireless connections and uses multiple network topologies, including point-to-point (P2P) topology for one-to-one (1:1) device communications. Bluetooth LE P2P optimizes data transfers and is ideal for connected device products, such as fitness trackers and health monitors.

Atracsys did chose to interface the spryTrack over Bluetooth low energy for the following reasons:

- The Bluetooth low energy stack is simpler than the Bluetooth classic stack. All Bluetooth low energy applications are GATT-based. This basically means that the data organization and exchange are well-defined in the specification. This lowers the implementation complexity for the integrators.
- Some mobile manufacturers such as Apple requires Bluetooth classic peripheral to go through their own certification process in order to interface their tablet. This restriction doesn't apply to devices using Bluetooth low energy.
- Surgical operations might be long. While the spryTrack isn't battery-powered at the moment – future work may include a battery pack –, other devices connecting to the spryTrack might be, e.g. a tablet. Using Bluetooth low energy diminishes the battery usage of these devices.

The spryTrack only provides an interface through Bluetooth Low Energy. Standard Bluetooth is not supported.

Any wrong settings of the spryTrack, misuse of the host PC/Tablet/Phone BLE stack or misuse of the OS functionalities might cause wrong measurements causing patient injury. The integrator should perform tests during the development process to validate configured settings.

The throughput of the Bluetooth Low Energy protocol allows the spryTrack to stream the informations of 4 markers for each frame (See section 8.6.5). If the connected device supports Bluetooth Low Energy short latency (7.5 ms), the framerate should reach 54 Hz (The maximum supported by the spryTrack).

8.6.1 Overview of the protocol

The spryTrack uses the high-performance Bluetooth Low Energy protocol.
The spryTrack is responsible to manage connection events from a host (or central) device and offers a subset of device options. Moreover, it implements various security features to protect communication with the central device (*Man-In-The-Middle* protection, passkey and *Near-Field-Communication* pairing, bonding...).
The spryTrack is compatible and supports the following Bluetooth features: high-throughput 2 Mbps, Advertising Extensions and channel selection algorithm #2 (CSA #2).

The spryTrack is not compatible with Long range physical layer. The spryTrack is compatible with physical layer 2 Mbps.

Constant frame rate is guaranteed only in 2 Mbps BLE Mode.

The spryTrack latency must be d by the user. Due to BLE protocol or environmental conditions, frame reception may be delayed and multiples frames my be received at the same time. The user should evaluate the occurrence and associated risks of this situation.

The spryTrack position may have an impact on BLE packets reception. Due to bad device and host position, frame reception may be delayed or lost. The user should evaluate the occurrence and risks of this situation.

Using the Bluetooth Low Energy Protocol to connect to the spryTrack is fairly straightforward given a good Bluetooth Low Energy Central Device library. Because of the limited bandwidth of the BLE protocol, only markers data and processing information can be retrieved from the spryTrack. (See 8.6.5 Tracking characteristic)

The followings chapters will show how to configure the BLE module to receive markers data.

8.6.2 Connection and configuration

The following part will explain the connection and configuration procedure. spryTrack.

- Scan for devices and connect to the spryTrack. The spryTrack advertises with the prefix “sTk ” following by the serial number of the device (16 digits).

- Pair the device, please use the *Near-Field-Communication* feature to ensure *Man-In-The-Middle* protection. A PIN code is also avaible for minimum security connection.Once the pairing is done, the bonding procedure will start. If successful, the PIN code will not be asked during future connection request between this spryTrack and this BLE Host.

The spryTrack manufacturing default passkey is: `123456`. BLE passkey can be modified using sample `stk28_BluetoothSecurtityConnexionManagement`. (See Section 8.3)

It is the integrator's/customer's/user's responsibility to take all the safety means that are necessary to ensure the security level required by the application.

- Once connected, if the library and the central device you are developing on supports it, change the connection parameters to request the lowest latency allowed by the Bluetooth Low Energy protocol (7.5 ms).
- Request MTU of 247 bytes. The stack supports an optional symbol rate of 2 Msym/s, with bit rate of 2 Mb/s, which is referred to as LE 2M PHY. This LE 2M PHY feature has been introduced in the Bluetooth Core specification version 5.0. While using an Android device, depending on your OS version, you may not use low connection interval like 7.5 ms.

As an additional security measure, it is necessary to use the Connection Bluetooth option detailed in section 8.6.6, to start using the spryTrack.

8.6.3 Exposed services and characteristics

Services available

Standard services:

`Base UUID: 00000000-0000-1000-8000-00805F9B34FB`

Short UUID	Service Name	Description
0x180a	Device Information Service	Specific device information
0x1801	Generic Attribute Service	Services monitoring and notification. Not used.
0x1800	Generic Access Service	General information about the device

Atracsys services:
`Base UUID: 00000000-a8ea-a583-b14a-7759f296a620`

Short UUID	Service Name	Description
0x2500	Data streaming service	Atracsys service.

Services characteristics

Characteristics list for standard services are described in Table 8.6.
The Atracsys service is further described in the following section: 8.6.4

8.6.4 The Atracsys service

The Atracsys spryTrack Service (sTk service) is a vendor specific service with vendor specific UUID.
Atracsys spryTrack UUID Base: `xxxxxxxx-A8EA-A583-B14A-7759F296A620`
Where `xxxxxxxx` corresponds to the 32 bits short UUID of the service and the characteristics.
The Atracsys service has 3 characteristics described in Table 8.7

Service Name	Short Characteristic UUID	Characteristic Name	R/W
Device Information Service	0x2a24	Model Number	R
	0x2a25	Serial Number	
	0x2a26	Firmware Revision	
	0x2a27	Hardware Revision	
	0x2a28	Software Revision	
	0x2a29	Manufacturer Name	
	0x2a50	PNP ID Characteristic	
Generic Access Service	0x2a00	Device Name	R
	0x2a01	Appearance	
	0x2a04	Peripheral Preferred Connection Parameters	
	0x2aa6	Central Address Resolution	

Table 8.6: spryTrack standard services description.

Name	Atracsys spryTrack service	Tracking characteristic	Option management characteristic	spryTrack log characteristic
Short UUID	0xAE3B2500	0xAE3B2501	0xAE3B2502	0xAE3B2503
Max length	NA	247 Bytes	247 Bytes	247 Bytes
Readable	NA	True	True	True
Writable	NA	False	True	False
Notifying	None	True	False	True
Security requirement	NA	bonding	bonding	bonding

Table 8.7: Atracsys service description.

8.6.5 Tracking Characteristic

A frame structure is a structure containing the on-board processed information of a picture.
The size of the frame structure is variable and based on the number of markers tracked.
The spryTrack can track a maximum of 4 markers simultaneously.
In Bluetooth Low energy mode, only some processed information is sent to the user:

- The number of fiducials seen by each camera
- The frame acquisition information
- The errors triggered by the on-board processing
- The number of markers successfully reconstructed
- For each marker all the standard information of a marker

The full frame content is represented in the array Table 8.8. For details about fields, please refer to the Chapter 7.11 How to use the spryTrack.

Name	Size in Bytes	Description
`left2DFiducialsCount`	1	Count of fiducials seen in left picture
`right2DFiducialsCount`	1	Count of fiducials seen in right picture
`frameCounter`	4	Frame acquisition number
`frameTimestamp`	8	Frame acquisition timestamp
`left2DFiducialsErr`	4	`fTkErrors` for 2D fiducials process seen in left picture
`right2DFiducialsErr`	4	`fTkErrors` for 2D fiducials process seen in right picture
`threeDFiducialsErr`	4	`fTkErrors` for 3D fiducials reconstruction
`markersErr`	4	`fTkErrors` for markers reconstruction
`markerCount`	1	Number of markers tracked in the scene
`BLEMarker buffer`	0 to 160	`BLEMarker` data buffer. See Table 8.9
`tempBuffer`	16	values of internals temperature sensors
`accMegaEventType`	1	types of following accMega event.
`accMegaEventBuffer`	12	contents of accMega event.

Table 8.8: Tracking charateristic frame content.

`BLEMarker` type is a subset of the standard `fTkMarkers` data type. The content of a `BLEMarker` element is described in Table 8.9

8.6.6 Option Management Characteristic

The option Management Characteristic allow the user to configure the spryTrack. It manages two types of options: Bluetooth low energy module options and Device options. The Bluetooth low energy module options are tools to manage the connection, execute tasks and set Bluetooth mode unrelated to the configuration of the acquisition process.

Name	Size in Bytes	Description
`Id`	4	Marker tracking Id
`geometryId`	2	Geometric id, i.e. the unique id of the used geometry
`geometryMask`	2	Presence mask of fiducials expressed as their geometrical indices.
`registrationErrorMM`	4	Mean registration error (unit mm) .
`translationMM`	12	Translation vector (unit mm).
`rotationRodrigues`	12	Rotation matrix of the marker, in Rodrigues vectorial representation
`status`	4	Marker status information.

Table 8.9: `BLEMarker` type content.

Once the procedure described in 8.6.2 is done, the Option management characteristic should be use to:

- Connect to the spryTrack using `Connection` Bluetooth energy module options.
- Set user geometry using `Load Geometry` Bluetooth energy module options.
- Map device options name with Id using `List Option` Bluetooth energy module options.
- Set user device options following procedure described in 8.6.6.
- Enable tracking using `Start streaming` Bluetooth energy module options.
- Then user should extract informations of tracking characteritic as described in 8.6.5.

The option Management characteristic can manage Devices option using the write property of the GATT characteristic. **Data provided for a write access on Option Management Characteristic must be structured exactly as shown in this section.**

- For Bluetooth low energy module options: ID + value (if necessary)
- For Device Options: ID + payload length + request type + value (if necessary)

Bluetooth low energy module options

All Bluetooth low energy module options values are only one byte long. (See Table 8.10).

To load a geometry using Bluetooth Low energy communication, you need to provide a geometry file as described in Section 7.8.

These files need to be added in the assets of your Android project so that an instance of `BLEGeometry` can be created using android spryTrack spryTrack library (this example library can be found in spryTrack

Name	Id	Description	Value
`Connection`	0xF800	Force the BLE connection state for the sTk	0x00 to disconnect, 0x01 to connect.
`List Option`	0xF600	Return in spryTrack Log characteristic the list of all device option, with value, description, and permissions.	NA
`Streaming mode`	0xF900	Manage the send of BLE frame in the Tracking Characteristic	0x00 to disable, 0x01 to enable.
`Unset Geometry`	0xF700	Disable the tracking of a geometry ID sent in payload.	Byte array containing the geometry ID.
`Set Geometry`	0xF701	Enable the tracking of a geometry sent in payload.	Byte array containing the parsed geometry from a geometry file.

Table 8.10: Bluetooth low energy module options descriptions.

SDK).

To load a geometry using the BLE option `Load Geometry` you need to create a byte array containing the `Load geometry` option Id (`0xF701`) and a byte array return by the the method `toFtkGeometry()`. The method `toFtkGeometry()` is part of the class `BLEGeometry` of the spryTrack Android Library.

Device options

A device option packet is composed of: `option Id + payload length + request type + value`

- Options Id should be extracted by executing a List Option command (See Option descriptions presented in Section 8.1.1
- The Payload length corresponds to the number of bytes of the value, plus one byte of request type. It cannot exceed 245 Bytes.
- Request type code are listed in Table 8.11
- Value: value must be converted in hexadecimal and transmitted with less important byte first.

Device option Id must be written on exactly 2 bytes, no conversion is needed unless the Device Option Id is superior to `9999`. In that case the first two number will be converted in a hexadecimal letter. (example: option id `10002` must be formatted as `A002`). As well device options id inferior to `1000` must be granted extra 0 to be written on 2 Bytes.

To reduce the risks of bad device configuration due to previous host connection to the spryTrack, Atracsys strongly recommend verifying at the beginning of the user application that the spryTrack configuration (device options, etc.) matches the expected one.

Request Type	Code
Read	0x00
Read max	0x01
Read min	0x02
Read default	0x03
Write	0x04

Table 8.11: device options request type.

Each request over a device option will trigger a notification over the spryTrack Log Characteristic. (See 8.6.7) Additional information on how to manage a device option is available on the documentation of spryTrack library.

8.6.7 Sprytrack Log Characteristic

SpryTrack Log Characteristic is used to get back information such as options values, status and log messages from the spryTrack.

Device options log

Each request over a device option will trigger a notification over the SpryTrack Log characteristic. If the configuration of the device option has failed, one error codes is return. If the option write or read was successful, it return a success code and the value in case of a read.

Received data	Code description
`E0-01`	Error code: Write failed
`E0-02`	Error code: Read failed
`00-00`	Write success
`00-xx`	Read success + read value: `xx`

Table 8.12: Sprytrack Log Charactristic return code description.

Bluetooth low energy module options log

Bluetooth low energy module List options will send a series of string, each one corresponding to one option of the spryTrack. Each string contains the complete option data set such as its name, its id, its description, properties and values. `BLEOption` class instances from the spryTrack android library should be constructed from a parsing of these `string`.

8.6.8 Roles and Security

In the spryTrack configuration, the spryTrack device has a BLE peripheral role and the smartphone or tablet has a BLE central role.

BLE defined 2 states (advertising or connected) and 3 stages of connections between 2 devices:

- connected
- paired

- bonded

After 2 devices (1 peripheral and 1 central) are connected, they can pair and optionally bond. For the spryTrack Vendor service, the devices are expected to connect and pair *Out-Of-Band* using *Near-Field-Communication* to insure *Man-In-The-Middle* protection. The use of a static passkey is a weak security measure. This type of connection is not recommended for medical purposes.

Man in the middle (*Man-In-The-Middle*) protection is only insure when the pairing has been done using *Near-Field-Communication*.

The *Near-Field-Communication* option is not available for the spryTrack 300. Please contact Atracsys in case of question.

8.7 The spryTrack user tools

The Atracsys spryTrack *SDK* comes with a set of binary user tools allowing the user to interact with a spryTrack device outside of the *SDK* library.

8.7.1 spryTrack toolbox command line interface – stk_toolbox_cli

the `stk_toolbox_cli` is a command line interface that allows basic interaction with the spryTrack without enumerating the device via the *SDK*. It may be used to check if a spryTrack is connected and availaible on the Host PC, or to reboot the spryTrack. The `stk_toolbox_cli` functionnality are described in its helper. To show the toolbox help, run the following command:
On Windows, open a command prompt from within the bin folder in the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") and run:

```
stk_toolbox_cli.exe  -h
```

On Linux, open a shell and, from within the installation path of the *SDK* (usually located in the `INSTALL\PATH\bin` folder), run:

```
./stk_toolbox_cli.exe  -h
```

8.7.2 sTk firmware programmer – stk_programmer64

the `stk_programmer64` is a tool that allows the users to update or dump the firmware of a specific spryTrack device. The firmware zip file is available on Atracsys support website. A helper on how to use the `stk_programmer64` can be displayed with the following command:
On Windows, open a command prompt from within the bin folder in the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") and run:

```
stk_programmer64.exe -h
```

On Linux, open a shell and, from within the installation path of the *SDK* (usually located in the `INSTALL\PATH\bin` folder), run:

```
./stk_programmer64 -h
```

Updating the firmware

One functionality of the `stk_programmer64` is to update the device firmware. It will update the spryTrack's different memories and perform the necessary check validating the successfulness of the update. For more information on how to update your device please refer to the Section 5.6.

Never downgrade a spryTrack firmware without the authorization of Atracsys. Always check the compatibility matrix between the *SDK* release number and the spryTrack firmware release number provided by Atracsys before proceed to an update.

Dumping the firmware

The other functionality of the `stk_programmer64` is to dump the device firmware into a zip archive file.

In case the user is experimenting issues that cannot be resolved with the indication of the Chapter 15, this functionality is intended to provide its exact spryTrack configuration to Atracsys for debugging purposes.

Never use a downgraded firmware zip file to flash another spryTrack device.

To dump the firmware please run the following command:
On Windows, open a command prompt from within the bin folder in the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") and run:

```
stk_programmer64.exe -a d -f [pathToYourDestinationFolder] -s [SN]
```

On Linux, open a shell and, from within the installation path of the *SDK* (usually located in the `INSTALL\PATH\bin` folder), run:

```
./stk_programmer64 -a d -f [pathToYourDestinationFolder] -s [SN]
```

Where `[pathToYourDestinationFolder]` corresponds to the path in which you want the archive entitled `sTkProgrammerDumpArchive.zip` to be created and `[SN]` the serial number of the device you intend to read the memory from.

8.7.3 USB streaming test – stk_usbDataTester64

the `stk_usbDataTester64` is a tool that allows the users to test the USB connection quality between the spryTrack and the host PC. This test is independent of the device scene of view, it's a tool to help the user to verify the quality of the USB link.

Depending on several factors such as the USB card used, the number of USB devices connected to the computer or just the USB port itself, the transfer rate of data between the spryTrack and the computer can be impacted. This tester program intends to ensure that the minimal required data rate for the spryTrack streaming application (which is the most demanding data rate of the spryTrack) is met within the user current USB configuration. To use the USB data rate test program the user can use the following command:
On Windows, open a command prompt from within the bin folder in the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") and run:

```
stk_usbDataTester64.exe -s [SN]
```

On Linux, open a shell and, from within the installation path of the *SDK* (usually located in the `INSTALL\PATH\bin` folder), run:

```
./stk_usbDataTester64 -s [SN]
```

Where `[SN]` corresponds to the serial number of the device you intend to read the memory from. An example of USB streaming test output is given in Listing 8.6.

```
================================================================================
            stk_usbDataTester - sTk USB data rate and integrity test
================================================================================

This software is the property of Atracsys LLC.

Current stk_usbDataTester64.exe version: 2.0.0.21.

          ============================================================
          =               stk_usbDataTester - Results                =
          ============================================================

* Test duration: 30.00 s

* Failed transfers: 0 - 0.00 %

* Valid transfers:                  29669 - 100.00 %
* Valid transfers with data errors: 0 - 0.00 %

        - Bytes processed:        8778055680
        - Data errors detected:   0  - 0.00 %
        - Size errors detected:   0
        - Micro-transfers missed: 0

======> Data rate: 292.60 MB/s - 2.34 Gb/s

Tested on the current USB port of this PC:
The USB data rate is suitable for spryTrack operation.
```

Listing 8.6: Example of `stk_usbDataTester64` output.

8.7.4 Shock sensor report – stk_GetShockMonitoringData64

The shock reports are obtained via the `stk_GetShockMonitoringData64` tool. The process is in two steps: 1. The data is downloaded from the device; 2. The data is sent to Atracsys' processing server and the pdf report is sent by email.

To use this tool you need to provide a directory on which the raw data will be stored and an email address for the result :
On Windows, open a command prompt from within the bin folder in the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") and run:

```
stk_GetShockMonitoringData64.exe -o [out] -e [email]
```

On Linux: please make sure that `mono` is installed and added to the `PATH` before running the program. Then, open a shell and, from within the installation path of the *SDK* (usually located in the `INSTALL\PATH\bin` folder), run:

```
./stk_GetShockMonitoringData64 -o [out] -e [email]
```

Where `[out]` corresponds to the output directory (ensure that the user has read&write permission on this directory) and `[email]` is the email where the report should be sent.

An acknowledge of the report request should be sent by email in the next few minutes, and the report should be received in the next 30 min.

8.7.5 spryTrack internal log streaming test – stk_trouble_shooting64

the `stk_trouble_shooting64` is a tool that allows the users to retrieve the internal logs of the spryTrack. In case the user is experimenting issues that cannot be resolved with the indication of the Chapter 15, this functionality is intended to provide its exact spryTrack configuration to Atracsys for debugging purposes. On Windows, open a command prompt from within the bin folder in the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") and run:

```
stk_trouble_shooting64.exe  -o [out]
```

On Linux, open a shell and, from within the installation path of the *SDK* (usually located in the `INSTALL\PATH\bin` folder), run:

```
./stk_trouble_shooting64  -o [out]
```

Where `[out]`corresponds to the output directory (ensure that the user has read&write permission on this directory). Please send the output file to Atracsys support team for analysis.

8.8 Embedded processing and USB2.0

The computation of the markers and fiducials positions can be done either on the host PC (in the *SDK*) or on the embedded processor of the spryTrack. When the computation is done on the host PC, the images taken by the spryTrack have to be streamed through the USB connection. Two options are available in the *SDK* to control where the processing is done and what is streamed through USB:

- "Enable images sending": Whether the images are streamed through USB.
- "Enable embedded processing": Whether the positions are computed on the spryTrack or on the host PC.

When using the spryTrack through a USB2.0 connection, it is highly recommended to *enable the embedded processing* and to *toggle off the images sending*. Otherwise, the framerate may be degraded.

8.9 SDK configuration file

Since version 4.2.1 of the SDK, the `ftkInitExt` function allows to specify a *JSON* configuration file. This file currently allows to set custom value of options for a device (as obtained from the 'Save environment' option, see Section 8.1.4). This configuration file consists of optional `"enviroment"` object.
The `"enviroment"` object currently only supports version 1, as shown in Listing 8.7.

```
{
    "environment": // Optional "environment" object
    {
    "devices":
    {
            "0x110A1C1B1ADE5028":
            {
        "options": // Mandatory array of options
        [
            // ...
        ],
        "sdkVersion": "v4.5.1" // Mandatory SDK version
            }
    }
    }
}
```

Listing 8.7: Configuration file with ”environment” section version 1.

9 Language bindings

In this chapter are described the different bindings Atracsys makes available to the users.

The reference implementation is the distributed C / C++ API. The other language bindings are not subject to thorough testing, as the C / C++ do. Integrator is responsible for his software validation using the other language bindings.

9.1 Python wrapper

A Python™ wrapper is distributed, which has been tested with Python 3.7.2 (64 bits) on Windows and on Linux. The C++ / Python interface framework used is pybind11.

The wrapper allows to use *almost* all possibilities allowed by the spryTrack C++ API, meaning that the classes of the C++ API have just been wrapped in Python.

The Python wrapper is distributed through a compressed archive in the python folder of the SDK. The wrapper needs Python (64 bits), git, CMake 3.10 or newer, and a C++11-compliant compiler (i.e. Microsoft Visual Studio 2015 or newer, gcc 5.0 or newer, clang 3.8 or newer).

To install the wrapper, just run `pip install archiveName.tar.gz` from the command line in the python directory of the SDK.

9.2 Matlab wrapper

A Matlab® wrapper is distributed, which has been tested with Matlab R2016a and Microsoft Visual Studio 2015 on Windows. It consists of a set of C++ source files, which are then compiled via Matlab using Visual Studio, and Matlab scripts files. The Matlab main script, provided as an example, uses the `FusionTrack` class, which is fully documented, as well as the underlying `FusionTrack` class written in C++.
The wrapper allows to:

- detected connected device(s);
- get and set `int32` and `float32` options;
- get `ftkBuffer` options;
- register / clear geometries;
- get the latest frame, with
  - *marker* data;
  - *fiducial* data;

- timestamp, counter;
- image size.

The Matlab wrapper is distributed as a source package containing a CMakeLists.txt file. CMake is responsible to get the Matlab installation and libraries, so that the correct flags are passed to the compiler. The wrapper needs Matlab (64 bits), a C++11-compliant compiler (i.e. Microsoft Visual Studio 2015 or newer, gcc 5.0 or newer, clang 3.8 or newer) compatible with the installed Matlab version.

In order to compile the wrapper on Windows, the cmake generation must be set in 64 bits (See Listing 9.1), and the build type must be specified on Unices (See Listing 9.2).

```
REM In a temporary folder do:
cmake -DCMAKE_GENERATOR_PLATFORM=x64 "C:\Program Files\Atracsys\fusionTrack SDK x64\matlab"
cmake --build . --config Release
```

Listing 9.1: Compiling the Matlab wrapper in windows.

```
# In a temporary folder do:
cmake -DCMAKE_BUILD_TYPE=Release $ATRACSYS_SDK_HOME/matlab
cmake --build .
```

Listing 9.2: Compiling the Matlab wrapper on unices.

On Listing 9.3 is presented how to initialise the wrapper, enumerate connected devices and get the SDK version. A geometry is then registered and finally, a frame is retrieved, showing 4 fiducials and a marker.

```
>> trSystem = FusionTrack()

trSystem =

  FusionTrack with properties:

    FTK_OPT_DRIVER_VER: 4
           FTK_MIN_VAL: 0
           FTK_MAX_VAL: 1
           FTK_DEF_VAL: 2
             FTK_VALUE: 3
               version: 1

>> serials = trSystem.devices()

serials =

 11889503043542755368

>> trSystem.getData( serials( 1 ), trSystem.FTK_OPT_DRIVER_VER )

ans =

v4.3.1 (1548418737) Windows-6.3.9600

>> geom = loadGeometry( 'C:\Program Files\Atracsys\fusionTrack SDK x64\data\geometry004.ini'
    );
>> trSystem.setGeometry( serials( 1 ), geom )
>> frame = trSystem.getlastFrame( serials( 1 ) )

frame =

```

```
32     threeDFiducials: [4x1 struct]
33             markers: [1x1 struct]
34         imageHeader: [1x1 struct]
```

Listing 9.3: Using the Matlab wrapper.

10 Control the operability of the spryTrack system and instruments

In this chapter is explained how to guaranty the operability of the spryTrack. Outputs and alert messages are generated to help protect against hazardous behaviours. They can also be used to validate the spryTrack operability.

It is assumed that the spryTrack is correctly setup as described in Chapter 5 and the software on the host PC is correctly installed as indicated in Chapter 6.

The various checks, presented in the following sections, allow to ensure that the computed *marker* position is valid. Different outputs and messages could be used to indicate the status of those tests.

10.1 Control spryTrack operability

The zeroing method must be performed before the navigation procedure is started, or during use, as soon as a marker is changed, cleaned, or if the spryTrack device must be cleaned, whilst the accuracy assessment method is performed with a frequency determined by the integrator.

There are three ways to check the spryTrack system is correctly working:

1. the monitoring of the registration error, see Section 10.2;
2. the ‘zeroing method’, described in Chapter 11;
3. the ‘accuracy assessment method’, described in Chapter 12.

The two last methods guarantee the spryTrack is operational in the configured environment, by checking the calibration, integrator settings and environment configuration. The zeroing method must be performed before the navigation procedure is started, or during use, as soon as a *marker* is changed, cleaned, or if the spryTrack device must be cleaned, whilst the accuracy assessment method is performed with a frequency determined by the integrator. The two methods allow to avoid (non-exhaustive list):

- using a decalibrated spryTrack device, which can be due to shock during transport, storing or handling;
- using the spryTrack device in noisy environment, such as an air perturbation between spryTrack device and *marker*;
- any electromagnetic interference with internal electronics;

all of those might lead to wrong measurements.

10.2 Control marker registration and epipolar errors

At any time, the marker registration error and the marker epipolar error (defined in Section 7.9) computed by the *SDK* must be monitored. The integrator must define thresholds above which the errors can lead to unacceptable risk for the patient. If the errors go above the aforementioned thresholds, the integrator must prevent the user from accessing the data.

A large epipolar error affecting all fiducials can be caused by the spryTrack operating outside its temperature range or being decalibrated.

A large registration error can be caused by the orientation of the *marker*: e.g. if the *marker* becomes close to parallel to the optical axis. The integrator shall generate an error message if the orientation goes above a defined angular threshold.

A large registration error can be caused by an object in the field of view which occludes partially the marker. The integrator shall state in their user manual that no objects in the field of view are allowed but the *markers*, pointers or surgical tools.

A large registration error can be caused by a scratched fiducial, liquids on fiducial, damaged fiducial, wrongly fixed fiducials on the marker, or a deformed marker. The integrator must generate an error message when the marker registration error is too high.

10.3 Control spryTrack temperature

The integrator shall monitor temperature sensors. If the measured temperatures diverge from the reference temperatures, the data shall not be given to the end-user.

The spryTrack device is operational after warm up. If used too early, during thermal expansion, this could lead to wrong measurements. The most accurate measurements are obtained when the temperature of the spryTrack corresponds to its calibration temperature. The *SDK* allows to access the spryTrack device temperature sensors. This is explained in details in the `sTk13_TemperatureMonitor`sample, whose code and binaries are available in the *SDK* installer / tarball. The calibration temperature is reached after approximately 30 min for a room temperature of 20 °C.

A too high temperature due to internal overheat, or due to the spryTrack device being wrongly placed near a heat source, might lead to unexpected temperature measurements. The integrator has to trigger an error message indicating that the spryTrack sensor has detected an abnormal temperature within the system.

Using the spryTrack device out of humidity specifications might lead to wrong measurements. The integrator has to specify in the end user manual the spryTrack humidity specifications as described in Table 5.1.

10.4 Control optical elements and IR illumination

Any degradation of optical elements due to the cleaning method or scratches during transport can lead to wrong measurements. Optical elements should be handled with care during transport and storage. The integrator has to mention in the end user manual the cleaning methods for optical elements as described in Chapter 14.

10.5 Control environmental conditions

Instability of the spryTrack device can cause injury by falling on patient or operator. Attachment between feet and holding mechanism must be perfectly stable. Instructions indicating how to fixate spryTrack using the threads in the feet are provided in Section 5.3.

Polluting light in the field of view might lead to wrong measurements. The integrator can access the camera images to visualize the scene of view. A segmentation overflow mechanism and status flags in the frame structures can be used by the integrator to detect any perturbation. The integrator is suggested to periodically check the camera pictures.

The spryTrack is not intended to be sterilised. A contamination on cameras could lead to patient contamination. The spryTrack has to be placed outside of the sterile field and depending on the intervention, cleaned before every single use. Methods of cleaning are indicated in Chapter 14.

The integrator must check that the spryTrack device and the position of the spryTrack device is well-suited to the application / surgery, and that required regulations are respected.

10.6 Integrator software

At any time, the hardware (i.e. the spryTrack device), the firmware and the software (i.e. the *SDK*) must be compatible. New releases of firmware, software and their respective compatibilty will be notified to the integrator.

A software dysfunction due to software crash might cause wrong measurements causing patient injury. The integrator has to indicate in the end user manual that the spryTrack device must be restarted when the software freezes or crashes.

A software dysfunction such as frozen pictures might cause wrong measurements causing patient injury. The integrator has to implement a software mechanism to detect such cases (e.g. check timestamps, etc.).

A software dysfunction such as frozen timestamp in pictures might cause wrong measurements causing patient injury. The integrator has to implement a software mechanism to detect such cases (e.g. check timestamps, etc.).

A software dysfunction such as memory leak may lead to frame corruption. The integrator has to implement a software mechanism to reduce the risks associate to such case (e.g. software filter, frame interpolation, etc.).

A software / hardware dysfunction such as frozen temperature modules might cause wrong measurements causing patient injury. The integrator has to implement a software mechanism to detect such cases (e.g. check temperatures values in frames, etc.).

In the case of the spryTrack device stops functioning during use / operation, it is the integrator responsibility to ensure that surgeon resume operation with the conventional method.

Any wrong settings of the spryTrack device, misuse of the SDK or misuse of the OS functionalities might cause wrong measurements causing patient injury. The integrator should perform unit tests during the development process to validate configured settings.

The integrator must take care that the host PC meets the SDK *and* their software requirement, especially the needed memory and the nominal CPU usage must be paid attention to.

The integrator implementation may cause inconsistent latency. The integrator should evaluate the system's latency once implemented and integrated in his product.

The integrator should perform tests during development process to verify system functionalities for all possible failure scenarii of its application

A software dysfunction due to unavailable data could lead to wrong measurements or the non performing of the surgery as planned. The integrator has to perform unit tests according to the end application of the spryTrack device.

A non respect of the software functionality or unexpected software failure might cause wrong measurements. The integrator should perform unit tests during the development process to validate the software functionality.

On Unix/Linux, the integrator must ensure the software is functional with the used GNU C library, C++ Standard Library and POSIX thread versions.

The integrator must generate a 'noisy environment' error message when the environment is poluted with reflective areas or other light sources.

The version of the *SDK* can be retrieved using the `ftkVersion` function, documented in [1].

11 The zeroing method

The zeroing method described in Chapter 11 must be implemented by the integrator and be automatically run at the application startup and inform the user on how to proceed.

The zeroing method is a procedure which checks that the spryTrack device and the used *markers* are correctly calibrated. A failure of this test does not allow to determine which element is faulty: it is a global test. In case of a failure of this test, the navigation process is not allowed to start.

11.1 General concept

The principle of the zeroing method is simple: a known distance is checked to be measured correctly. Most surgical applications use a pointer, the position of the pointer tip is therefore a good candidate to be monitored. This is achieved by using the pointer and an additional *marker*, the latter is required to have a cavity, named "divot", in which the tip of the pointer must fit whatever the pointer position and orientation (e.g. a cone or a cylinder), as illustrated in Figure 11.1. Once the pointer tip has been inserted in the *marker* cavity, the pointer should be rotated around its tip, which must remain in the *marker* cavity. The spryTrack records the position and orientation of the marker and the pointer, from which the position of the tip of the pointer relative to the marker is extracted. This relative position should not vary within measurement uncertainties. During this operation, the marker must lie around the centre of the working volume, as shown in Figure 11.2.

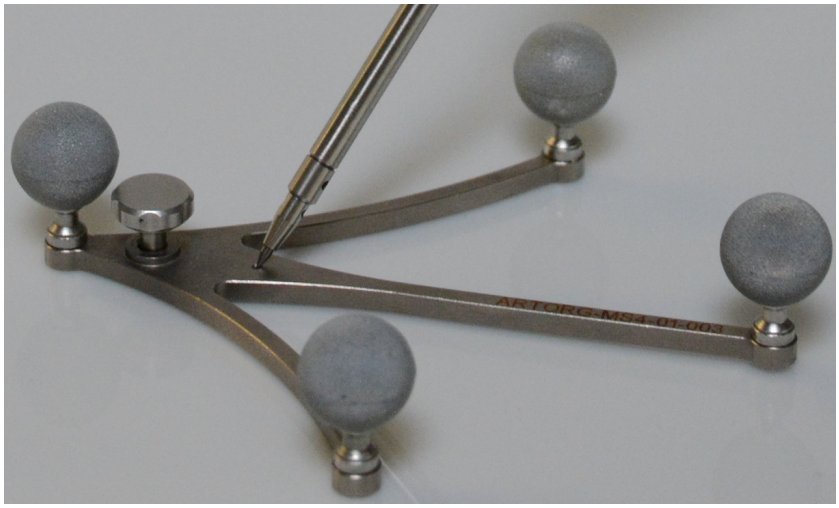


Figure 11.1: The tip of the pointer must lie in the cavity.

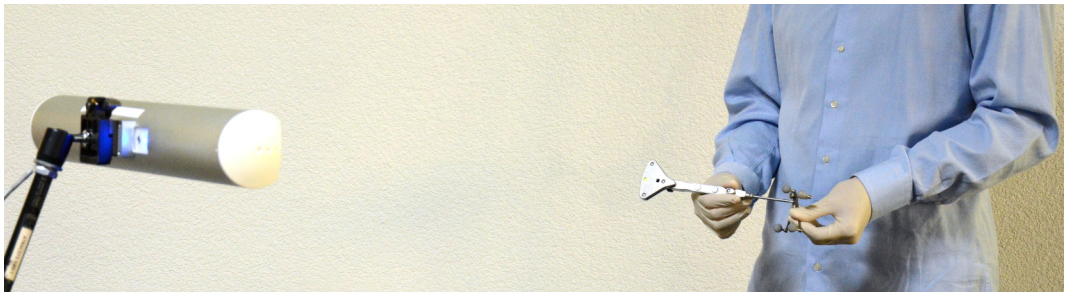


Figure 11.2: Running the zeroing procedure.

The integrator must specify a maximal value on the error of the position, above which the test is considered to fail. A failure of the test may be caused by (non-exhaustive list):

- a deformed pointer or *marker*;
- a decalibrated spryTrack device;
- dirty optics on the spryTrack device.

11.2 Running the procedure

The zeroing procedure must be integrated in the workflow at each startup of the application or as soon as the pointer (whole pointer or any of its *fiducials* in case of passive *markers*) is changed. The operator for this test is the surgeon. How to proceed must be clearly stated in the user manual.

	Any liquid on lenses such as body fluids ejection or condensation could lead to wrong measurements. Lenses have to be cleaned as soon as they get in contact with liquids. The zeroing method has to be re-performed after each cleaning.

	Some disposable passive *fiducials* cannot be cleaned, as the reflective coating would be damaged, and may require to be changed during the operation. The zeroing method has to be re-performed after changing any *fiducials*.

12 The accuracy verification tool (AVT)

Accuracy verification must be implemented by the integrator. They can optionnaly purchase the required software and hardware at Atracsys.

The operator performing accuracy verification must be correctly trained.

The accuracy verification method is a procedure which checks that the spryTrack device is able to perform accurate measurements in the whole working volume. A failure of this test indicates that the spryTrack device must not be used and must be returned to Atracsys.

12.1 General concept

The principle of this test is to verify that a given rigid body keeps its fixed geometry wherever the measurement is performed. For a *marker*, the distance between two *fiducials* is determined and cannot change, independently of the *marker*'s position in the working volume. The test consists of moving a *marker* in the whole working volume, the best way to ensure repeatability being to define 3D positions at which the *marker* should stay for a while (e.g. for 500 frames) and monitor the *marker*'s registration error as well as the distances between the *marker*'s *fiducials*. If the used *marker* is accuratly calibrated, it may be enough to look at the width of the distribution of the registration error. The integrator must specify an upper bound on this width, above which the test is considered to fail. In general, monitoring the distance between two *fiducials* of the *marker* is a suitable way to perform this test: again the width of the distribution of the measured length must not be larger than a threshold specified by the integrator.

12.2 Running the procedure

This procedure is not part of the usual running of the *navigation system*: it must be run by a trained technician, on a regular basis. For instance the integrator can choose to run this procedure once every week or every 5 surgeries, whatever comes first.

12.3 The accuracy verification tool from Atracsys

Atracsys offers as an aditional accessory a rigid body designed to perform accuracy verification on the spryTrack. Along with this rigid body, Atracsys provides the required software to perform the capture of data and the accuracy analysis. Please contact the Atracsys after sale service for further information (See Section 1.6).

13 Electromagnetic compatibility

The spryTrack is a medical electrical equipment and needs special precautions regarding EMC and needs to be installed and put into service according to EMC information provided in this document.

EMC tests have included the *Power Injector* for the USB mode tests.

Applied standards	
EN 60601-1-2:2007 +AC:2010 IEC 60601-1-2:2007 (ed3.0)	Medical electrical equipment – Part 1-2: General requirements for basic safety and essential performance – Collateral standard: Electromagnetic compatibility – Requirements and tests
EN 60601-1-2:2015 IEC 60601-1-2:2014 (ed4.0)	Medical electrical equipment – Part 1-2: General requirements for basic safety and essential performance – Collateral standard: Electromagnetic compatibility – Requirements and tests
EN 60601-1-2:2015/A1:2021 IEC 60601-1-2:2020 (ed4.1)	Medical electrical equipment – Part 1-2: General requirements for basic safety and essential performance – Collateral standard: Electromagnetic compatibility – Requirements and tests

Electromagnetic emissions		
The spryTrack is intended for use in the electromagnetic environment specified below. The integrator, the customer or the user should assure that it is used in such an environment.		
Emissions test	Compliance	Electromagnetic environment - guidance
RF emissions EN 55011:2016 CISPR 11:2015 (ed6.0)	Group 1	**While functionning in USB mode** the spryTrack uses RF energy only for its internal function. Therefore, its RF emissions are very low and are not likely to cause any interference in nearby electronic equipment.
RF emissions EN 55011:2016 CISPR 11:2015 (ed6.0)	Group 2	**While functionning in BLE mode** the spryTrack must emit electromagnetic energy in order to perform its intendedfunction. Nearby electronic equipment may be affected.
RF emissions EN 55011:2016 CISPR 11:2015 (ed6.0)	Class B	The spryTrack is suitable for use in all establishments, including domestic establishments and those directly connected to the public low-voltage power supply network that supplies buildings used for domestic purposes.
Harmonic emissions EN 61000-3-2:2014 IEC 61000-3-2:2014	Not applicable	
Voltage fluctuations / Flicker emissions EN 61000-3-3:2013 IEC 61000-3-3:2013	Not applicable	

Use of the spryTrack adjacent to, or stacked with other equipment should be avoided because it could result in improper operation and can lead to wrong measurements to serious injury including death. If such use is necessary, the spryTrack and the other equipment should be observed to verify that they are operating normally.

Electromagnetic immunity			
The spryTrack is intended for use in the electromagnetic environment specified below. The integrator, the customer or the user should assure that it is used in such an environment.			
Immunity test	Test level	Compliance level	Electromagnetic environment - guidance
Electrostatic discharges EN 61000-4-2:2009 IEC 61000-4-2:2008	±8 kV contact ±15 kV air	±8 kV contact ±15 kV air	Floors should be wood, concrete or ceramic tile. If floors are covered with synthetic material, the relative humidity should be at least 30 %.
Electrical fast transient / burst EN 61000-4-4:2012 IEC 61000-4-4:2012	±2 kV for power supply lines ±1 kV for input/output lines	±2 kV for power supply lines ±1 kV for input/output lines	Mains power quality should be that of a typical commercial or hospital environment.
Surge EN 61000-4-5:2014 IEC 61000-4-5:2014	±1 kV line to line ±2 kV line to earth	±1 kV line to line no earth connected	Mains power quality should be that of a typical commercial or hospital environment.
Voltage dips, short interruptions and voltage variations on power supply input lines EN 61000-4-11:2004 IEC 61000-4-11:2004	< 5 % Ut (> 95 % dip in Ut) for 0.5 cycle 40 % Ut (60 % dip in Ut) for 5 cycles 70 % Ut (30 % dip in Ut) for 25 cycles < 5 % Ut (> 95 % dip in Ut) for 5 s	< 5 % Ut (> 95 % dip in Ut) for 0.5 cycle 40 % Ut (60 % dip in Ut) for 5 cycles 70 % Ut (30 % dip in Ut) for 25 cycles < 5 % Ut (> 95 % dip in Ut) for 5 s	
Power frequency (50/60Hz) magnetic field EN 61000-4-8:2010 IEC 61000-4-8:2009	30 A m^{-1} 50 Hz and 60 Hz	30 A m^{-1} 50 Hz and 60 Hz	Power frequency magnetic fields should be at levels characteristic of a typical location in a typical commercial or hospital environment.
Conducted RF EN 61000-4-6:2014 IEC 61000-4-6:2013	3 V_{RMS} 150 kHz to 80 MHz 10 V_{RMS} ISM and Amateur bands	10 V_{RMS} 150 kHz to 80 MHz 10 V_{RMS} ISM and Amateur bands	Portable and mobile RF communications equipment should be used no closer to any part of the spryTrack, including cables, than the recommended separation distance calculated from the equation applicable to the frequency of the transmitter. Recommended separation distance $d = 0.3\sqrt{P}$ 150 kHz to 80 MHz $d = 0.35\sqrt{P}$ 80 MHz to 800 MHz $d = 0.70\sqrt{P}$ 800 MHz to 2.7 GHz Where (P) is the maximum output power rating of the transmitter in watts (W) according to the transmitter manufacturer and d is the recommended separation distance in metres (m). Field strengths from fixed RF transmitters, as determined by an electromagnetic site survey[a], should be less than the compliance level in each frequency range.[b] Interference may occur in the vicinity of equipment marked with the following symbol: 
Radiated RF EN 61000-4-3:2006 +A1 +A2 IEC 61000-4-3:2006 +A1 +A2	3 V m^{-1} 80 MHz to 2.7 GHz	10 V m^{-1} 80 MHz to 3.0 GHz	

a The ISM (industrial, scientific and medical) bands between 150 kHz and 80 MHz are 6,765 MHz to 6,795 MHz; 13,553 MHz to 13,567 MHz; 26,957 MHz to 27,283 MHz; and 40,66 MHz to 40,70 MHz.

b The compliance levels in the ISM frequency bands between 150 kHz and 80 MHz and in the frequency range 80 MHz to 2,5 GHz are intended to decrease the likelihood that mobile/portable communications equipment could cause interference if it is inadvertently brought into patient areas. For this reason, an additional factor of 10/3 has been incorporated into the formulae used in calculating the recommended separation distance for transmitters in these frequency ranges.

c Field strengths from fixed transmitters, such as base stations for radio (cellular/cordless) telephones and land mobile radios, amateur radio, AM and FM radio broadcast and TV broadcast cannot be predicted theoretically with accuracy. To assess the electromagnetic environment due to fixed RF transmitters, an electromagnetic site survey should be considered. If the measured field strength in the location in which the spryTrack is used exceeds the applicable RF compliance level above, the spryTrack should be observed to verify normal operation. If abnormal performance is observed, additional measures may be necessary, such as reorienting or relocating the spryTrack.

d Over the frequency range 150 kHz to 80 MHz, field strengths should be less than 3 V m^{-1}.

Table 13.1: Electromagnetic immunity test.

Proximity fields From RF wireless communications equipment					
Test frequency (MHz)	Band (MHz)	Modulation	Test Level	Compliance Level	Observations
385	380 to 390	Pulse, 18 Hz	27 $V m^{-1}$	27 $V m^{-1}$	
450	430 to 470	FM, ±5 kHz	28 $V m^{-1}$	28 $V m^{-1}$	
710 745 780	704 to 787	Pulse, 217 Hz	9 $V m^{-1}$	9 $V m^{-1}$	
810 870 930	800 to 960	Pulse, 18 Hz	28 $V m^{-1}$	28 $V m^{-1}$	
1720 1845 1970	1700 to 1990	Pulse, 217 Hz	28 $V m^{-1}$	28 $V m^{-1}$	
2450	2400 to 2570	Pulse, 217 Hz	28 $V m^{-1}$	28 $V m^{-1}$	Losts packets in BLE mode
5240 5500 5785	5100 to 5800	Pulse, 217 Hz	9 $V m^{-1}$	9 $V m^{-1}$	

Table 13.2: Proximity fields from RF wireless communication equipment immunity test.

Use of accessories, transducers and cables other than those specified or provided by Atracsys could result in increased electromagnetic emissions or decreased electromagnetic immunity of this equipment and result in improper operation and/or loss of performance. Such failure can lead to wrong measurements, serious injury including death.

Portable radio frequency communications equipment (including peripherals such as antenna cables and external antennas) should be used no closer than 30 cm (12 inches) to any part of the spryTrack, including cables specified by Atracsys. Otherwise, degradation of the performance of this equipment could result, which can lead to wrong measurements, serious injury including death.

spryTrack with structered light projectors While using the spryTrack with dot-pattern projectors, some configurations are to be considered to guarantee the functionalities of the device. The power supply provided with the *Power Injector* is sensitive to burst perturbations requiring some constraints on the number of dot-pattern projectors enabled and the duration of the strobe

The limitation depends on the burst perturbation that needs to be considered in your region.

For regions following IEC 60601-1-2:2014, edition 4.0 requiring burst perturbations of 100 kHz at 100 V and 240 V, the following configurations need to be used:

- 1 point projectors enable: 16000 us of strobe time
- 2 point projectors enable: 500 us of strobe time
- 3 point projectors enable: 400 us of strobe time

Separation distances between portable and mobile RF communications equipment and the spryTrack according to frequency of transmitter (m)				
The spryTrack is intended for use in an environment in which radiated RF disturbances are controlled. The integrator, the customer or the user of the device can help prevent electromagnetic interference by maintaining at least the minimum, or the recommended distance between portable and mobile RF communications equipment (transmitters) and the spryTrack as recommended below, according to the maximum output power of the communications equipment.				
Rated maximum output power of transmitter (W).	Minimal separation distance (m)	Recommended separation distance (m)		
	$d = 0.6\sqrt{P}$	150 kHz to 80 MHz $d = 1.2\sqrt{P}$	80 MHz to 800 MHz $d = 1.2\sqrt{P}$	800 MHz to 2.7 GHz $d = 2.3\sqrt{P}$
0.01	0.06	0.12	0.12	0.23
0.1	0.19	0.38	0.38	0.73
1	0.60	1.20	1.20	2.30
10	1.90	3.80	3.80	7.27
100	6.00	12.00	12.00	23.00

Table 13.3: Required separation Distance between RF equipment and the spryTrack.

For China only, following IEC 60601-1-2:2014, edition 3.0 requiring burst perturbations of 5 kHz and 100 kHz, the following configurations need to be used:

- 1 point projectors enable: 1500 us of strobe time
- 2 point projectors enable: 500 us of strobe time
- 3 point projectors enable: 400 us of strobe time

With these configurations the functionalities of the spryTrack are guaranteed in case of a burst perturbation with our power supply. However, it is possible to use the device at full power (3 point projectors and 16000 us of strobe time) but is as the own risk of the integrator, the customer or the user. Atracsys LLC disclaims all liability.

13.1 Essential performances

As the spryTrack device is only a component for a medical device, the essential performances of the overall medical device have to be defined and validated by the legal manufacture of the medical device.

13.1.1 Quality of the measurement

The precision, accuracy and trueness of the position measurement of the spryTrack device depend on a multitude of factors such as fiducial type (LED, sphere, sticker), marker geometry, working distance, partial occlusion and many more. In the typical setup that was tested, the quality of the measurement was not affected by any of the electromagnetic immunity tests stated in Table 13. To supervise this immunity, the parameters shown in Table 13.4 have been evaluated during the electromagnetic immunity tests.

Evaluation parameter	Value
Blob number	Constant
Number of fiducials used to build a marker	Constant
Marker registration error	Constant [a]
Registration error max - min	Constant [a]
Fiducial interdistance	Constant [a]
Fiducial interdistance max - min	Constant [a]

[a] With presence of a jitter, which depends on: *fiducial* type (LED, sphere, sticker), *marker* geometry, working distance.

Table 13.4

13.1.2 Measurement rate

The rate of which the measurements are delivered by the spryTrack device on the host PC depends on a multitude of factors such as number and size of blobs (white zones) in the images, USB speed, host PC, OS, blocked optical path and many more. In the typical setup that was tested, the measurement rate was not affected by any of the electromagnetic immunity tests stated in the Table 13 except for the following cases Table 13.5:

Immunity test	Note
Radiated RF	The BLE connection may be perturbed by radiated field with a frequency between 2.3 GHz and 2.5 GHz. This may results into a temporary acquisition frames rates drop.[a]

[a] Only if the spryTrack is used in BLE mode.

Table 13.5

In BLE mode the spryTrack device operates in the unlicensed ISM band at 2.4 GHz. In case the spryTrack device is used around the other wireless devices which operate in similar frequency band (2.3 GHz to 2.5 GHz) of the spryTrack device, Interference mays occures and produce packets losts in BLE communication. If such interference occurs, the acquisition frame rate is not guaranted.

Portable radio frequency communications equipment (including peripherals such as antenna cables and external antennas) should be used no closer than 30 cm (12 inches) to any part of the spryTrack device, including cables specified by Atracsys. Otherwise, degradation of the performance of this equipment could result, which can lead to wrong measurements, serious injury including death.

For the supervision of the measurement rate, the following parameters have been evaluated during the electromagnetic immunity tests.

Evaluation parameter	Value
Acquisition rate	Constant $>$ 50 Hz [a]
Dropped frames rate	Constant [a]
Processed frames rate	Constant [a]

[a] Those values may depend on the test environment and/or host PC.

14 Maintenance

In this chapter are found the instructions for maintaining basic safety and performances for the expected service life, in particular it is explained how to clean the spryTrack device and its accessories.

14.1 Cleaning the spryTrack device

The spryTrack was not designed to be autoclavable. The end user manual instruction shall provide a validated cleaning and disinfection method to the end-user.

The spryTrack can be disinfected with regular hospital disinfectant detergent applied using a clean cloth. The optical parts must be cleaned only using camera lenses cleaning solutions. Eyeglass lens tissues must not be used as it may scratch the lens. Other surfaces must be dried using a clean and dry cloth.

The device must be turned off and disconnected prior to cleaning. The disinfecting detergent must not be spilled directly onto the device or any of its components.

It must be ensured that no fluids enter the spryTrack. This may damage the spryTrack by causing short-circuits and corrode the material. Do not immerge the spryTrack in liquid.

Only non-corrosive and non-acidic detergents, whose interactions with material are known, may be used.

14.2 Sterilisation of the markers

The integrator must ensure markers are not deformed after sterilisation.

The battery of an active marker must be properly removed and disposed of before sterilisation.

14.3 Effects of multiple cleanings

The spryTrack optics may be scratched or the window transparency may be altered, leading to poor quality picture and therefore to poor tracking accuracy. The loss of tracking accuracy can be detected using the procedure described in Chapter 12. In the case of a device not passing the test, it should not be used anymore and sent back to Atracsys for repair.

14.4 Disposal of equipment

To ensure environmentally responsible disposal decommissioned equipment, please contact Atracsys, see Section 1.6.

15 Troubleshooting

This chapter addresses the main problems a user can experience when using the spryTrack system.

15.1 Boot sequence

When plugging the spryTrack to a correct power source (see section 3.2) such as the spryTrack Power Injector, the following sequence should be observable:

- The status LED is blinking green.
- The blinking green LED turned fixed green with a buzzer sound.
- The spryTrack device can correctly be detected by your operating system via USB or via BLE (see Section 15.2).

If the above sequence is not observable, please find below the boot known issues that can occur:

15.2 spryTrack recognition by the Operating System

+Upon a successful boot of the device, the spryTrack is ready to be used and should be detectable by the Host (USB or BLE host). If there is no device detected by your system. Or if the system detects a spryTrack entitled “Atracsys spryTrack FLM” and/or with a null serial number (“0x0000000000000000”), ensure that, if any firmware update has recently been performed, you have correctly followed the firmware update procedure (see Section 5.6). If that is the case, contact Atracsys (see Section 1.6).

15.2.1 USB connection mode

Windows

USB devices are listed in the ”Device Manager” utility. To view them, click on the start menu button and search for ”Device Manager”. Once the program is opened, expand the node ”Universal Serial Bus devices”. (See Figure 15.1). The spryTrack should be displayed there with the label ”Atracsys spryTrack”.

While the USB connexion speed of the device may not be easy to see in the ”Device Manager” panel, Microsoft developed a graphical tool that is included in the Windows Driver Kit (WDK).

Problem	Suggested solution
All LEDs are off when power on the spryTrack.	- Check that the used cables are not damaged in any way. - Check that a correct source is used to power the device or check the spryTrack Power Injector. - Check the *Power Injector* blue LED is on and the red LED is off. - Use other *USB-C* and USB cable combinations (see Section 3.2.2).
The spryTrack is beeping and the status LED is blinking in red.	- Check that you are using a valid USB Power-Delivery power source within the spryTrack specifications (see Section 3.2.1). - Check the *Power Injector* blue LED is on and the red LED is off. - If any firmware update has recently been performed refers to Section 5.6 or contact Atracsys (see Section 1.6).
+ The device cannot be detected by the host PC.	- Check the USB cables; - Unplug and replug the device
The firmware update fails in the middle of the operation, the demo application is unstable.	- Unplug as many as possible USB devices from the host computer. - Unplug and replug the device. - If possible, try to use another USB port on the host computer.
The spryTrack device is beeping and the status LED is blinking in red.	Contact Atracsys (see Section 1.6).

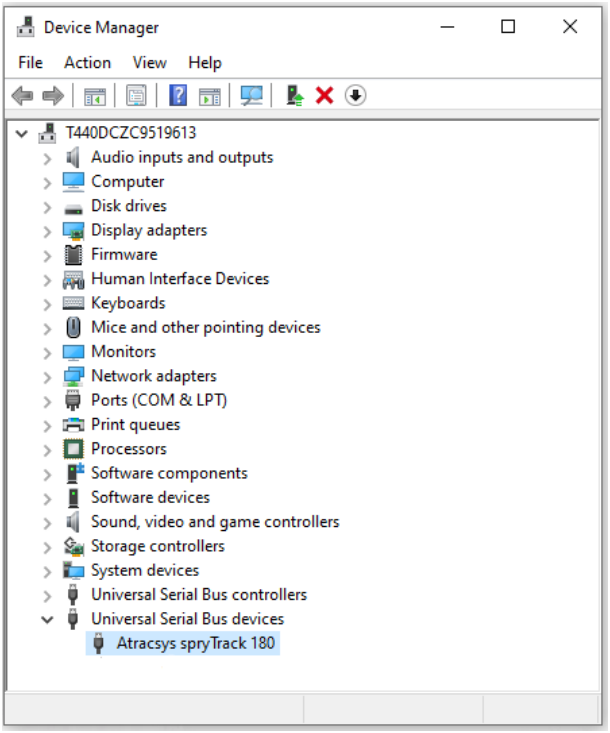


Figure 15.1: The node Atracsys spryTrack 180 is detected correctly.

15.2.2 Linux

On Linux, USB devices can be listed using the command "lsusb -v". If an entry with the same device descriptor as in Listing 15.1 appears in the list, the spryTrack is recognized.

While using a Linux OS, please follow the installation procedure described in Section 6.2 before. Otherwise the spryTrack may not be detected.

```
Bus 002 Device 003: ID 1c82:0200
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               3.00
  bDeviceClass          255 Vendor Specific Class
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         9
  idVendor           0x1c82
  idProduct          0x0200
  bcdDevice            0.01
  iManufacturer           1 Atracsys LLC
  iProduct                2 Atracsys spryTrack 180
  iSerial                 3 0xBA000009214F5C28
```

Listing 15.1: Output of lsusb entry for the spryTrack 180.

15.2.3 BLE connection mode

This section will treat BLE related issues. Before any check, please verify that the boot sequence described in Section 15.1 is correct, and verify that the host BLE device is compatible with the spryTrack.

Advertising

After a successful boot sequence, the spryTrack should advertise with its serial number and the prefix `sTk_-` followed by the spryTrack serial number. Use your BLE compatible device by trying to pair a new Bluetooth device to verify the good advertising of the spryTrack. If the device is not advertising, please check that:

- There is no other Host (BLE or USB) already connected to the spryTrack.
- The device option `NFC Only` is not activated (please see the SDK example `stk28_BluetoothSecurtityConnexion` in section Section 8.3)

Near-Field-Communication only is a protected advertising mode. If it is activated, the status LED is blue during the boot sequence (see Section 15.1). To ensure that *Near-Field-Communication* only mode is disabled, reboot the device and check that at the end of the boot sequence, the LED is green (not blue).

To manage the advertising mode please run the SDK example `stk28_BluetoothSecurtityConnexionManagement`. (See section Section 8.3).

Passkey management

The *SDK* sample `stk28_BluetoothSecurtityConnexionManagement` described in section Section 8.3 allows to read and modify the BLE passkey of the spryTrack. It may also be used to enable or disable the *Near-Field-Communication* only mode.
If your passkey is valid and the pairing is not successful, please contact Atracsys.

Option and streaming

In case of issues managing option and/or streaming through Bluetooth Low Energy, we highly recommend that you follow again the procedure described in section 8.6.6.
If the issue persists, please contact Atracsys.

15.3 Software *SDK* use

To be able to use the Atracsys SDK the spryTrack needs to have correctly booted and needs to be connected to a USB port with a USB 3.0 connection speed or to a USB-C port with those capabilities.
If you are not able to use the software, please check if your issue is listed in Table 15.1.

To see *SDK* log message:

- On Windows, please run the `Logger64` program present in the installation path of the *SDK* (usually "C:\Program Files\Atracsys\spryTrack SDK\bin") before launching your application.
- On Linux, *SDK* logs are automatically written in a text file at the end of the application run.

Problem	Suggested solution
Incapacity of enumerating the spryTrack and the following log doesn't appear: "devStkHelpers.cpp:26: Found device matching Atracsys LLC VID 0x1c82 and spryTrack PID 0x0200." spryTrack.	Check that the spryTrack device is correctly recognized by your operating system (see Section 15.2.1).
Incapacity of enumerating the device with the log: "devStkHelpers.cpp:64: The spryTrack is seen with an USB 2.0 speed or lower: 0x210. Performances will be degraded."	- Check that you are correctly connected to a USB3 port. - Check that, if you are using the spryTrack Power Injector, you have correctly followed its connection recommendation and indication (see Section 3.2.1). - See Section 15.4 libusb backend.
Incapacity of enumerating the device with the log: "spryTrack.cpp:115: Detected libusbK, aborting."	See Section 15.4 libusb backend.
The software seems to be stuck at enumeration, without any specific message.	See Section 15.2.1 USB.
spryTrack frame drop or incapacity of enumerating the device.	Verify your USB conexion performance with the tool stk_streamer. (Please refers to section 8.7.3 for details).

Table 15.1

15.4 libusb backend

The Atracsys *SDK* uses the open source library “libusb”. This library provides a multi-platform support for USB operations. It relies on “backend” for each platform to perform the low-level operations. On Windows, it uses the native Microsoft backend. However, it can also use an open source backend called “libusbK” which is incompatible with the USB3 transfers used by the spryTrack *SDK*. Unfortunately, this “libusbK” backend is prioritized by libusb instead of the native Microsoft backend. If this backend is detected on your Windows operating system, the *SDK* will refuse to operate and an error message will be displayed: “spryTrack.cpp:115: Detected libusbK, aborting.”. You need to uninstall libusbK and all of its traces from your computer so that the native Microsoft backend can be used by the *SDK*.

15.5 Retrieving the last logs of a running system

The tool stk_trouble_shooting.exe can be used to retrieve the last system logs of a spryTrack (See Section 8.7).

15.6 Sending an image of the system to Atracsys for analysis

The tool stk_programmer.exe can be used to retrieve the spryTrack firmware and then send to Atracsys for analysis. (See Section 8.7).

16 Licences

This chapter enumerates the various software used by Atracsys spryTrack SDK, the demo software and the *AVT* software.

16.1 Atracsys spryTrack SDK headers and library

These files are part of the Atracsys spryTrack library. You can freely use this SDK for your own applications.

A license to use binary files of this distribution is granted to owner of infiniTrack, spryTrack and/or fusionTrack devices.

It is not allowed to redistribute any source file of this distribution without written permission of Atracsys.

16.2 Hashing code

The spryTrack SDK uses hashing code by Stephan Brumme.

Unless otherwise noted in a file's first 5 lines, all source code published on http://create.stephan-brumme.com and its sub-pages is licensed similar to the zlib license:

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software.
2. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
3. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

16.3 About JsonCpp

The spryTrack SDK uses JsonCpp library to parse JSON files.

The JsonCpp library's source code, including accompanying documentation, tests and demonstration applications, are licensed under the following conditions…

Baptiste Lepilleur and The JsonCpp Authors explicitly disclaim copyright in all jurisdictions which recognize such a disclaimer. In such jurisdictions, this software is released into the Public Domain.

In jurisdictions which do not recognize Public Domain property (e.g. Germany as of 2010), this software is Copyright © 2007-2010 by Baptiste Lepilleur and The JsonCpp Authors, and is released under the terms of the MIT License (see below).

In jurisdictions which recognize Public Domain property, the user of this software may choose to accept it either as 1) Public Domain, 2) under the conditions of the MIT License (see below), or 3) under the terms of dual Public Domain/MIT License conditions described here, as they choose.

The MIT License is about as close to Public Domain as a license can get, and is described in clear, concise terms at:

http://en.wikipedia.org/wiki/MIT_License

The full text of the MIT License follows:

==
Copyright ©2007-2010 Baptiste Lepilleur and The JsonCpp Authors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the ”Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED ”AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
==
(END LICENSE TEXT)

The MIT license is compatible with both the GPL and commercial software, affording one all of the rights of Public Domain with the minor nuisance of being required to keep the above copyright notice and license text in the source code. Note also that by accepting the Public Domain ”license” you can re-license your copy using whatever license you like.

16.4 About Qt

The demo software uses Qt.

Qt is a C++ toolkit for cross-platform application development.

Qt provides single-source protability across MS Windows, Mac OS X, Linux, and all major commercial Unix variants. Qt is also available for embedded devices as Qt for Embedded Linux and Qt for Windows CE.

Qt it available undre three different licensing options designed to accommodate the needs of our various users.

Qt licensed under our commercial licence agreeement is appropriate for develpment of proprietary/commercial software where you do not want to share any source fode with third parties or otherwise cannot comply with the terms of the GNU LGPL verison2.1 or GNU GPL version 3.0.

Qt licensed under the GNU LGPL version2.1 is appropriate for the develpment of Qt applications (proprietary or open source) provided you can comply with the terms and conditions of the GNU LGPL version 2.1.

Qt licensed under the GNU General Public Licences version 3.0 is appropriate for the develpment of Qt applications where you whish to use such applications in combination with software subject to the terms of the GNU GPL version 3.0 or where you are otherwise willing to comply with the terms of the GNU GPL version 3.0.

Please see `qt.nokia.com/product/licensing` for an overview of Qt licensing.

16.5 About QDarkStyle

The demo software uses QDarkStyle.

16.5.1 The MIT License (MIT) - Code

16.5.2 Creative Commons Attribution International 4.0 - Images

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

Considerations for licensors: Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. More considerations for licensors.

Considerations for the public: By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor's permission is not necessary for any reason–for example, because of any applicable exception or limitation to copyright–then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. More considerations for the public.

16.5.3 Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 - Definitions

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope

a: **License grant**.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:
   A. reproduce and Share the Licensed Material, in whole or in part; and
   B. produce, reproduce, and Share Adapted Material.
2. **Exceptions and Limitations**. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.
3. **Term**. The term of this Public License is specified in Section 6(a).
4. **Media and formats; technical modifications allowed**. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.
5. **Downstream recipients**.
   A. **Offer from the Licensor – Licensed Material**. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.
   B. **No downstream restrictions**. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. **No endorsement**. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b: **Other rights**.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.
2. Patent and trademark rights are not licensed under this Public License.
3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. **Attribution**.

1. If You Share the Licensed Material (including in modified form), You must:
   A. retain the following if it is supplied by the Licensor with the Licensed Material:
      i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);
      ii. a copyright notice;
      iii. a notice that refers to this Public License;
      iv. a notice that refers to the disclaimer of warranties;
      v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
   B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
   C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.
2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.
4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. **Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.**

b. **To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.**

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License termnate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

   1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
   2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the “Licensor.” Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark “Creative Commons” or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.
Creative Commons may be contacted at creativecommons.org.

16.6 About FreeGLUT

The demo software uses FreeGLUT.

FreeGLUT is an open source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT (and hence FreeGLUT) allows the user to create and manage windows containing OpenGL contexts on a wide range

of platforms and also read the mouse, keyboard and joystick functions. FreeGLUT is intended to be a full replacement for GLUT, and has only a few differences.

Since GLUT has gone into stagnation, FreeGLUT is in development to improve the toolkit. It is released under the MIT License.

16.7 About Coin3D

The demo software uses Coin3D.

Coin3D is a high-level, retained-mode toolkit for effective 3D graphics development. It is API compatible with Open Inventor 2.1.

Coin3D is Free Software, published under the BSD 3-clause license.

16.8 Base 64

The demo software uses base 64 encoding by René Nyffenegger.

This source code is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this source code must not be misrepresented; you must not claim that you wrote the original source code. If you use this source code in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original source code.
3. This notice may not be removed or altered from any source distribution.

René Nyffenegger rene.nyffenegger@adp-gmbh.ch

16.9 About FLTK

The *AVT* software is based in part on the work of the FLTK project (`http://www.fltk.org`) version 1.3.3.

December 11, 2001
The FLTK library and included programs are provided under the terms of the GNU Library General Public License (LGPL) Version 2, June 1991 with the following exceptions:

1. Modifications to the FLTK configure script, config header file, and makefiles by themselves to support a specific platform do not constitute a modified or derivative work.
2. The authors do request that such modifications be contributed to the FLTK project - send all contributions through the "Software Trouble Report" on the following page: http://www.fltk.org/str.php
3. Widgets that are subclassed from FLTK widgets do not constitute a derivative work.

4. Static linking of applications and widgets to the FLTK library does not constitute a derivative work and does not require the author to provide source code for the application or widget, use the shared FLTK libraries, or link their applications or widgets against a user-supplied version of FLTK.
5. If you link the application or widget to a modified version of FLTK, then the changes to FLTK must be provided under the terms of the LGPL in sections 1, 2, and 4.
6. You do not have to provide a copy of the FLTK license with programs that are linked to the FLTK library, nor do you have to identify the FLTK license in your program or documentation as required by section 6 of the LGPL.

However, programs must still identify their use of FLTK. The following example statement can be included in user documentation to satisfy this requirement:

> [*program/widget*] is based in part on the work of the FLTK project (`http://www.fltk.org`).

A Mathematical considerations

A.1 Homogeneous coordinates

In mathematics, homogeneous coordinates[3] for a 3-dimensional space uses 4-dimensional vectors to represent a position:

$$\vec{x} = \begin{pmatrix} x_0 & x_1 & x_2 \end{pmatrix}^T \iff X = \begin{pmatrix} x_0 & x_1 & x_2 & 1 \end{pmatrix}^T. \tag{A.1}$$

This notation allows a much easier handling of transformations composed of a rotation and a translation, as the transformation matrix is written as:

$$T = \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \tag{A.2}$$

with

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix}$$

and

$$\vec{t} = \begin{pmatrix} t_0 \\ t_1 \\ t_2 \end{pmatrix}$$

From those definitions result the combination of two transformations T_1 and T_2 result in a transformation $T_{tot} = T_2 \cdot T_1$.

A.2 Marker pose

Atracsys gives access to the marker orientation, either via the `ftkMarker::rotation` or the `atracsys::MarkerData::_Rotation` fields. The `ftkMarker::rotation` matrix is stored such that it is accessed like this: `ftkMarker::rotation[rowId][columnId]`, and the information is simply copied into the `atracsys::MarkerData::_Rotation` field (i.e. indices are used in the same way). Consider the following example, taken from the simulator data, in which a marker with geometry ID 2 is reconstructed with a registration error of 344 µm. The given rotation and translation are:

$$R = \begin{pmatrix} 0.5880 & -0.8023 & 0.1029 \\ -0.7987 & -0.5960 & -0.0828 \\ 0.1278 & -0.0336 & -0.9912 \end{pmatrix} \qquad \vec{t} = \begin{pmatrix} 199.319 \\ -103.765 \\ 876.320 \end{pmatrix}$$

The rotation components are retrieved from the SDK as presented on Listing A.1. Geometry ID 2 defines the following four points:

$$\vec{p_0} = \begin{pmatrix} 0.000 & 11.000 & 3.000 \end{pmatrix}^T \qquad \vec{p_1} = \begin{pmatrix} 24.090 & -15.750 & 3.000 \end{pmatrix}^T$$
$$\vec{p_2} = \begin{pmatrix} 0.000 & -47.690 & 3.000 \end{pmatrix}^T \qquad \vec{p_3} = \begin{pmatrix} 23.630 & -10.580 & 3.000 \end{pmatrix}^T$$

```
// C API example: framePtr is a ftkFrameQuery pointer
r12 = framePtr->markers[ 0u ].rotation[ 1u ][ 2u ]; // this is -0.0828
// C++ API example: frame is a atracsys::FrameData instance
r21 = frame.markers().front().rotation().at( 2u ).at( 1u ); // this is -0.0336
```

Listing A.1: Access to the rotation components, from both C and C++ API, same R and $\vec{t}$ as above.

The corresponding reconstructed 3D points are:

$$\vec{x_0} = \begin{pmatrix}191.153 & -110.391 & 873.010\end{pmatrix}^T \qquad \vec{x_1} = \begin{pmatrix}198.294 & -75.553 & 870.821\end{pmatrix}^T$$
$$\vec{x_2} = \begin{pmatrix}237.517 & -75.823 & 874.934\end{pmatrix}^T \qquad \vec{x_3} = \begin{pmatrix}221.837 & -116.360 & 876.675\end{pmatrix}^T$$

Now, using homogeneous coordinates, it can be checked easily that:

$$\begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p_i} \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x_i} \\ 1 \end{pmatrix} \quad \forall i \in [0;3]$$

$$\begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p_0} \\ 1 \end{pmatrix} = \begin{pmatrix} 190.802 \\ -110.569 \\ 872.977 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x_0} \\ 1 \end{pmatrix} \qquad \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p_1} \\ 1 \end{pmatrix} = \begin{pmatrix} 198.099 \\ -75.386 \\ 870.775 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x_1} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p_2} \\ 1 \end{pmatrix} = \begin{pmatrix} 237.889 \\ -75.902 \\ 874.949 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x_2} \\ 1 \end{pmatrix} \qquad \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{p_3} \\ 1 \end{pmatrix} = \begin{pmatrix} 222.010 \\ -116.581 \\ 876.743 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} \vec{x_3} \\ 1 \end{pmatrix}$$

A.3 Inverting a marker pose

If the position and orientation of a marker is needed in the coordinate system of another marker, the transformation should be inverted. When homogeneous coordinates are used, this invertion is trivial:

$$X_{\text{tracker}} = T \cdot X_{\text{marker}}$$
$$X_{\text{marker}} = T^{-1} \cdot X_{\text{tracker}}$$

If usual 3-dimensional coordinates are used, the invertion reads:

$$\vec{x}_{\text{tracker}} = R \cdot \vec{x}_{\text{marker}} + \vec{t}$$
$$\vec{x}_{\text{marker}} = R^{-1} \cdot \vec{x}_{\text{tracker}} - R^{-1} \cdot \vec{t}$$

As R is a rotation matrix, R^{-1} can be trivially obtained using $R^{-1} = R^T$. A common mistake is to use $-\vec{t}$ instead of $-R^{-1} \cdot \vec{t}$.

References

[1] Atracsys LLC, *Atracsys Passive SDK API documentation (doxygen documentation)*.

[2] http://en.wikipedia.org/wiki/INI_file, 2014. Online; accessed 2015-01-13.

[3] A. F. Möbius, *Der barycentrische Calcül: ein neues Hilfsmittel zur analytischen Behandlung der Geometrie*. 1827.