ART
Advanced Realtime Tracking

DTrack®SDK Programmer's Guide

How to Build and Run Examples

Version 2.8.0
January 2023

Trademarks

The following overview shows the registered trademarks of Advanced Realtime Tracking GmbH & Co. KG:

trademarks	illustrated as	**in Germany**	**in the EU**	**in the U.S.**
A.R.T.®	***ART***	×		
ART®	***ART***	×	×	×
ARTtrack®	***ARTTRACK***	×	×	×
DTrack®	***DTRACK3***	×		
smARTtrack®	***SMARTTRACK***	×		×
ART Advanced Realtime Tracking		×	×	×
ART Advanced Realtime Tracking		×	×	×

Warranty regarding further use of open source software

Advanced Realtime Tracking GmbH & Co. KG (denoted as ***ART***) provides no warranty for the open source software programs contained in this product, if such programs are used in any manner other than the program execution intended by ***ART*** . The licenses define the warranty, if any, from the authors or licensors of the open source software. ***ART*** specifically disclaims any warranties for defects caused by altering any open source software program or the product's configuration. You have no warranty claims against ***ART*** in the event that the open source software infringes the intellectual property rights of a third party. Technical support, if any, will only be provided for unmodified software. The full license text can be obtained from ***ART*** directly.

Disclaimer

In no event shall Advanced Realtime Tracking GmbH & Co. KG be liable for any incidental, indirect, or consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use the software or hardware.

ART
Advanced Realtime Tracking

Advanced Realtime Tracking GmbH & Co. KG
Am Öferl 6
82362 Weilheim i. OB
Germany
☎ +49 (0)881-92530-00
+49 (0)881-92530-01
https://ar-tracking.com

1 DTrackSDK

Purpose of DTrackSDK

A set of functions to provide an interface to ***ART*** tracking systems. It supports DTrack(1), DTrack2 and DTRACK3.

The functions receive and process DTrack measurement data packets (UDP; ASCII), and send/exchange DTrack/DTrack2/DTRACK3 command strings (UDP/TCP; ASCII).

License

This library is distributed under the BSD 3-clause License. You can modify the sources and/or include them into own software (for details see “license.txt”).

How to receive and process ***ART*** tracking data

DTrack/DTrack2/DTRACK3 uses ethernet (UDP/IP datagrams) to send measurement data to other applications. They both use an ASCII data format.

In its most simple operating mode DTrackSDK is just receiving and processing these data. In this case DTrackSDK just needs to know the port number where the data are arriving; all necessary settings have to be done manually in the DTrack/DTrack2/DTRACK3 frontend software.

DTrack/DTrack2/DTRACK3 also provides a way to control the tracking system through a command interface via ethernet. Both DTrack and DTrack2/DTRACK3 use ASCII command strings. DTrack2/DTRACK3 commands are sent via a TCP/IP connection.

DTrackSDK offers several C++ constructors to cover the various operating modes; prefer the “universal constructor” to obtain the best flexibility.

The formats and all other necessary definitions are described in “DTrack2 User Manual, Technical Appendix” or “DTRACK3 Programmer’s Guide”.

Sample source codes for an own interface

The sample source code files show how to use the DTrackSDK:

example_universal.cpp	sample using the universal constructor for all modes
example_listening.cpp	sample for pure listening
example_listening_multicast.cpp	sample for pure multicast listening
example_communicating.cpp	sample to control a DTrack2/DTRACK3 Controller
example_fingertracking.cpp	***ART*** FINGERTRACKING sample
example_tactile_flystick.cpp	sample using a Flystick to control a tactile FINGERTRACKING device
example_flystick_feedback.cpp	sample to control a Flystick with feedback
example_dtrack1.cpp	sample to control a DTrack(1) PC

Each example uses a different constructor and explains how to use it.
All examples are written in C++, and work for both Unix and Windows. The files have been successfully tested under Linux, Windows 7 and Windows 10.

For Windows: please link with library “ws2_32.lib”.
For Windows: example_tactile_flystick.cpp needs to be linked with “winmm.lib”.
For older Linux: example_tactile_flystick.cpp needs to be linked with “-lrt” (only for glibc versions before 2.17).

Compatibility with legacy SDK versions

DTrackSDK comes with additional sources that should make it easier to upgrade from older SDK versions. Actually they are just wrappers around class “DTrackSDK”, and allow to use the known interface. They are available for the classes:

DTracklib	class esp. for DTrack(1) systems (C++; used from May 2005 until Dec 2006)
DTrack	class esp. for DTrack(1) systems (C++; used from Jan 2007 until Dec 2010)
DTrack2	class esp. for DTrack2 systems (C++; used from May 2008 until Dec 2010)

Note that the wrappers just work for the C++ SDK versions; there’s no wrapper for the “DTracklib” SDK written in C.

Source Documentation (HTML)

Please refer to “./doc/html/index.html”.

Contents of this package

license.txt	copy of BSD 3-clause License
include/	C++ headers
src/	C++ implementations
examples/	C++ sample sources (see above)
Compatibility/	C++ wrappers for compatibility with legacy SDK classes
doc/html/	HTML documentation generated with doxygen (main file is “index.html”)

Development with DTrackSDK

a) Create a new project using the DTrackSDK:

- include the header file:
  include/DTrackSDK.hpp
- add following include paths:
  include/
- add following source files to your project:
  src/DTrackSDK.cpp
  src/DTrackData.cpp
  src/DTrackNet.cpp
  src/DTrackParse.cpp
  src/DTrackParser.cpp
- you may want to start with one of the example files provided in this package
- if you're unsure which C++ constructor to use best, take the "universal constructor" (like shown in "example_universal.cpp"); it provides the best flexibility

b) Upgrade an existing project developed with legacy DTrack SDK versions:

- upgrade from class "DTracklib":
  - include the header file:
    Compatibility/DTracklib/DTracklib.hpp
  - add following include paths:
    include/
    Compatibility/DTracklib/
  - add following source files to your project:
    src/DTrackSDK.cpp
    src/DTrackData.cpp
    src/DTrackNet.cpp
    src/DTrackParse.cpp
    src/DTrackParser.cpp
    Compatibility/DTracklib/DTracklib.cpp
- upgrade from class "DTrack":
  - include the header file:
    Compatibility/DTrack/DTrack.hpp
  - add following include paths:
    include/
    Compatibility/DTrack/
  - add following source files to your project:
    src/DTrackSDK.cpp
    src/DTrackData.cpp
    src/DTrackNet.cpp
    src/DTrackParse.cpp

src/DTrackParser.cpp
Compatibility/DTrack/DTrack.cpp

- upgrade from class “DTrack2”:
  - include the header file:
    Compatibility/DTrack/DTrack2.hpp
  - add following include paths:
    include/
    Compatibility/DTrack2/
  - add following source files to your project:
    src/DTrackSDK.cpp
    src/DTrackData.cpp
    src/DTrackNet.cpp
    src/DTrackParse.cpp
    src/DTrackParser.cpp
    Compatibility/DTrack2/DTrack2.cpp

2 Detailed Instructions

2.1 How to build and run examples on Windows

Used in this example

- ART Tracking System (ATC/ARTTRACK/TRACKPACK System or SMARTTRACK/SMARTTRACK3)
- Windows 10 64-bit
- Visual Studio 2017 Express Edition
- DTrackSDK_v2.8.0.zip

DTrack2/DTRACK3 settings

- One or more bodies are calibrated and tracked
- Output settings: Send to → this computer
  Port 5000 (make sure that your system's firewall doesn't block this port)

How to build an example

1. Create an empty project
2. Copy following files from "DTrackSDK_v2.8.0.zip" to your project folder
   a) complete "include" folder
   b) complete "src" folder
   c) "examples/example_universal.cpp"
3. Include the files to the project in Visual Studio
   Project → Add Existing Item...
   (Take care that the project is selected)
4. Project → Properties → Configuration Properties → Linker → Input
   include at "Additional Dependencies": ws2_32.lib and press enter
5. Project → Properties → Configuration Properties → Linker → System
   set "SubSystem" to: Console (/SUBSYSTEM:CONSOLE)

6. Project → Properties → Configuration Properties → VC++ Directories → Include Directories → <Edit...> → New Line (Ctrl-Insert) → Browse... → Point to the "DTrackSDK/include" path in DTrackSDK installation

   Steps 4, 5 & 6 have to be done in debug and release mode

7. Build the example

How to use

1. Start the measurement of the ART Tracking System
2. Start the program in the windows console: example_universal.exe 5000
3. Tracking data are displayed in the windows console

The other examples can be build and run analog to this example

2.2 How to build and run examples on Linux

Used in this example

- ART Tracking System (ATC/ARTTRACK/TRACKPACK System or SMARTTRACK/SMARTTRACK3)
- openSUSE 15.4 64-bit
- GNU Compiler Collection g++ 7.5.0
- DTrackSDK_v2.8.0.zip

DTrack2/DTRACK3 settings

- One or more bodies are calibrated and tracked
- Output settings: Send to → this computer
  Port 5000 (make sure that your system's firewall doesn't block this port)

How to build an example

1. Unzip DTrackSDK_v2.8.0.zip
2. Copy following files from "DTrackSDK_v2.8.0/" to your project folder
   a) complete "include" folder
   b) complete "src" folder
   c) "examples/example_universal.cpp"
3. Create a new file named "Makefile" in the project folder
4. Copy and paste the following code into the makefile (including the tabs and the new line like displayed below) and save it

all: \
[TAB] examples/example_universal

%: %.cpp
[TAB] g++ -o $* -I include/ $< src/*.cpp

[TAB] means: Change all indentiations from white spaces to tabs

5. Run `$ make` in the terminal

How to use

1. Start the measurement of the ART Tracking System
2. Run the terminal and change to the examples directory
3. Start the program in the terminal: `./example_universal 5000`
4. Tracking data are displayed in the terminal

The other examples can be build and run analog to this example