

## INSTALLATION GUIDE

# Leddar™ SDK

## **Software Development Kit**

## **Using LeddarTech Sensors in Windows**

© 2021 LeddarTech Inc. All rights reserved. This document may not be reproduced, distributed or disclosed, in whole or in part, without the express written consent of LeddarTech.

## **Table of Contents**

1.	LEDD	DARSDK FOR WINDOWS	3
	1.1. Bui	ILDING THE SDK IN WINDOWS	3
	1.1.1.	Windows C++ Package	3
	1.1.2.	Windows Python Package	
2.	INST	ALLING ROS FOR WINDOWS	19
3.	RUN	NING LEDDAR SENSORS IN ROS FOR WINDOWS	20
	3.1. Ins	TALLING LEDDARPY AND NUMPY IN ROS	20
	3.1.1.	Building LeddarPy	20
:	3.2. Pri	EPARING THE CATKIN WORKSPACE	21
:	3.3. Ru	NNING LEDDAR PIXELL WITH ROS FOR WINDOWS	23
	3.3.1.	Leddar Pixell ROS Code Examples	24
:	3.4. Ru	NNING LEDDAR VU8 WITH ROS FOR WINDOWS	
	3.4.1.	Leddar Vu8 ROS Code Examples	
:	3.5. Ru	NNING LEDDAR M16 WITH ROS FOR WINDOWS	
	3.5.1.	Leddar M16/IS16 ROS Code Examples	
	3.6. Ru	NNING LEDDAR T16 WITH ROS FOR WINDOWS	
	3.6.1.	Leddar T16 ROS Code Examples	
		1	

### 1. LeddarSDK for Windows

Download the latest version of the SDK from https://github.com/leddartech/LeddarSDK.

Extract the LeddarSDK-master folder.

For the purposes of this Installation Guide, the LeddarSDK-master folder has been created in the **c:\opt** directory for enhanced ROS compatibility.

* > This PC > Windows (C:) > opt > LeddarSDK-master				
Name	Date modified	Туре	Size	
📔 libs	2020-06-03 09:02 PM	File folder		
膧 src	2020-06-03 09:02 PM	File folder		
📄 .gitignore	2020-06-03 09:02 PM	Text Document	1 KB	
LeddarSDK.chm	2020-06-03 09:02 PM	Compiled HTML Help	19,094 KB	
LICENSE	2020-06-03 09:02 PM	File	2 KB	
👔 README.md	2020-06-03 09:02 PM	MD Document	3 KB	

### 1.1. Building the SDK in Windows

Before proceeding with the installation of any other package than Python, you first need to build the LeddarSDK solution.

Build process uses cmake (https://cmake.org). The **CMakeLists.txt** file is in the **src** folder. It is recommended to do an out-of-source build in the **src/release** folder.

You can choose for what sensor to build the SDK by modifying the cmake project configuration. All sensors are enabled by default.

### 1.1.1. Windows C++ Package

CMake is directly integrated in Visual Studio. Check the documentation here.

Open Visual Studio 2019 (16.8.5) and navigate to the folder containing a CMakeLists.txt file. In our case:

.\LeddarSDK-master\src

🔀 Visual Studio Community 2019

16.8.5

Open recent			G	et sta	irted		
Search recent (Alt+S)	- م			<b>*</b>	Clone or che	eck out code	
- C:\opt\LeddarSDK-ma	ster\src\LeddarConfigurator4				Get code from an c or Azure DevOps	online repository like GitHub	
LeddarPy.sln	stad eva) LaddaeDu	2020-12-03 10:08 AM		<b>?</b> ```	Open a proje	ect or solution	
Older	ster(src(LeddarPy			Ľ	Open a local Visual	Studio project or .sln file	
LeddarPy.vcxproj		2020-10-19 05:25 PM	_				
C:\opt\LeddarSDK-ma	ster\src\LeddarPy			2	Open a local	folder	
++ LeddarCDemo.vcpro	<b>j</b> mentr\LeddarTech\LeddarCDe	2020-08-28 04:01 PM			Navigate and edit of	code within any folder	
Examples da	ments (Leddar Tech (Leddar CDe	2020-08-26 11-18 AM					
C:\Users\user236\Docu	ments\LeddarTech	2020-00-20 11.10 AW		Ð	Create a new Choose a project te	v project emplate with code scaffolding	
LeddarConfigurator	4.sln	2020-06-11 12:15 PM			to get started		
C:\\LeedarTech\Modu	iles\SDK\LeddarSDK-master\sr	rc\LeddarConfigurator4			Continue	without code $\rightarrow$	
LeddarConfigurator	4_Prototypes.sln	2020-03-06 02:29 PM					
rowse  → • ↑ 🖹 «	LeddarSDK-maste	er > src >	~	Ū	Search src		
rowse → ▼ ↑ 💽 « anize ▼ New fo	LeddarSDK-mast	er > src >	~	Ū	Search src		
rowse → ✓ ↑ 💽 « anize ▼ New for Desktop	LeddarSDK-mastr older	er > src >	~	U	Search src	Date modified	
rowse → ▼ ↑ È « anize ▼ New fo Desktop Documents	LeddarSDK-mast	er > src >	~	Ū	Search src	■== ▼ Date modified 2021-01-25 10:25	5 AN
rowse → ✓ ↑ È « anize ▼ New fo Desktop Documents Documents Downloads	eddarSDK-mastro older Name Leddar	er > src >	~	U	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25	5 AN
rowse $\rightarrow$ $\checkmark$ $\uparrow$ $\blacktriangleright$ « anize $\checkmark$ New for $\models$ Desktop B Documents $\clubsuit$ Downloads $\clubsuit$ Music	LeddarSDK-mast older Name Leddar Ceddar	er > src > ^ ROS	~	Ū	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25	5 AN 5 AN 5 AN
rowse → ✓ ↑ È « anize ▼ New fo Desktop Documents Downloads Music Pictures	LeddarSDK-master older Name Leddar Leddar Leddar	er > src > ROS Example Py	¥	U	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25	5 AN 5 AN 5 AN 5 AN
rowse → ~ ↑ È « anize ~ New fo Desktop Documents Downloads Music Pictures Videos	eddarSDK-mastro older Name Leddar Eddar Leddar Leddar Leddar	er > src > ROS Example Py Fech	~	Ũ	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25	5 AN 5 AN 5 AN 5 AN 5 AN
rowse $\rightarrow$ $\checkmark$ $\uparrow$ $\blacktriangleright$ « anize $\checkmark$ New for Desktop Documents Downloads Music Pictures Videos Windows (C)	LeddarSDK-mast older Name Leddar LeddarF LeddarF LeddarT LeddarT P_ledd	er > src > ROS Example <sup>2</sup> y Fech lar_utils	~	Ŭ	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25	5 AN 5 AN 5 AN 5 AN 5 AN
rowse →	LeddarSDK-master Name Leddar Leddar Leddar LeddarT LeddarT Py_ledd Shared	er > src > ROS Example Py Fech lar_utils	~	Ū	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25	5 AN 5 AN 5 AN 5 AN 5 AN 5 AN
rowse	LeddarSDK-mast Name Leddar Leddar LeddarE LeddarB LeddarT Py_ledd Shared	er > src > ROS Example Py fech lar_utils	~	Ũ	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25	5 AN 5 AN 5 AN 5 AN 5 AN 5 AN
rowse → ✓ ↑ È « anize ▼ New for Desktop Documents Downloads Music Pictures Videos Videos Videos Unidows (C:) Intranet (I:) DFS_Server (\\lea	LeddarSDK-mast Name Leddar Leddar LeddarF LeddarF LeddarT Py_ledd Shared	er > src > ROS Example Py Fech lar_utils	~	ŭ	Search src	Date modified 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25 2021-01-25 10:25	5 AN 5 AN 5 AN 5 AN 5 AN 5 AN
rowse →	Vame Name Name Leddar Leddar Leddar Leddar Leddar Leddar Leddar R Leddar R Sata	er > src > A ROS Example Py Fech lar_utils	~	U	Search src	Date modified         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25         2021-01-25 10:25	5 AN 5 AN 5 AN 5 AN 5 AN 5 AN

Visual Studio will automatically detect the cmake structure, allowing you to build and debug the solution.





The SDK needs to be built in Release mode.

To do so, create a new configuration specifically for the release (e.g., x64-Release) and make sure to select the correct parameters.



	$\sim$	
<b>bug</b> ws (64-bit) with the MinGW development environment. MINGW_PREFIX must be set. (Debug)	1	
lease ws (64-bit) with the MinGW development environment. MINGW_PREFIX must be set. (ReIWith		
ws (32-bit) with the Visual Studio development environment. (Debug)	I	
ws (32-bit) with the Visual Studio development environment. (RelWithDebInfo)		
ws (64-bit) with the Visual Studio development environment. (Debug)		
ws (64-bit) with the Visual Studio development environment. (RelWithDebInfo)		
ebug dows (32-bit) with the Visual Studio development environment. (Debug)		
x86-Clang-Release Clang on Windows (32-bit) with the Visual Studio development environment. (RelWithDebInfo)		
Select Cancel	,	
Select Cancel		
Select         Cancel           to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings.json file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json.           Edit JSON	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings.json file. Add a new configuration to build and debug or eWindows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json.       Edit JSON         General       General	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettingsjson file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettingsjson.       Edit JSON         General       Configuration name:       A diate fine the configuration is to configuration to build and debug or defined in the underlying CMakeSettingsjson.	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettingsjson file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettingsjson.       Edit JSON         General       Configuration name: A Friendly name that identifies the configuration.       Af Kendense	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings.json file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json       Edit JSON         Edit JSON       General         Configuration name:       A friendly name that identifies the configuration.         x64-Release       Configuration type:	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings.json file. Add a new configuration to build and debug or e/Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json       Edit JSON         Edit JSON       General         Configuration name:       A friendly name that identifies the configuration.         x64-Release       Configuration type:         The build type to be used. "Debug" includes debug information in the compiled files to allow easy debugging. This corresponds to CMAKe SULD. TYPE.	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings.json file. Add a new configuration to build and debug or eVindows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json.       Edit JSON         Edit JSON         Configuration name:         A friendly name that identifies the configuration.         x64-Release       Configuration type:         The build type to be used. "Debug" includes debug information in the compiled files to allow easy debugging. This corresponds to CMAKE.BUID_TYPE.         RetWithDebinfo       *	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings.json file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json.       Edit JSON         General       Configuration name: A friendly name that identifies the configuration.       Kd4-Release         Configuration type:       The build type to be used. "Debug" includes debug information in the compiled files to allow easy debugging. This corresponds to CMAKE_BUILD_TYPE.         RetWithDebinto       *         Toolset:       *	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings json file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json.       Edit JSON         Configuration name:         A friendly name that identifies the configuration.       Image: Configuration name:       Image: Configuration name:         A friendly name that identifies the configuration.       Image: Configuration name:       Image: Configuration name:         Reverse       Configuration type:       Image: Configuration name:       Image: Configuration name:         A friendly name that identifies the configuration.       Image: Configuration name:       Image: Configuration name:         RewriteDebindo       Image: Configuration name:       Image: Configuration name:       Image: Configuration name:         RewriteDebindo       Image: Configuration name:       Image: Configuration name:       Image: Configuration name:         Totete:       Totete:       Totete:       Totete:       Totete:         Toteded environment above. This setting maps to inhertification contents in CMakeSettings.json.       Totese:       Totese:	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettingsjoon file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettingsjoon.       Edit JSON         General       Configuration name: A friendly name that identifies the configuration.       Kif4-Release         Configuration type:       The build type to be used. "Debug" includes debug information in the compiled files to allow easy debugging. This corresponds to CMAKE_BUID_TYPE.         ReWithDebinfo       *         Toolset:       Toolset:         Toolset:       Toolset:         msrc_x64_x64       *	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettingsjoon file. Add a new configuration to build and debug or eWindows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettingsjoon.       Edit JSON         Ceneral       Configuration name:       Atfiendly name that identifies the configuration.       Edit JSON         MarkeSettingsion       Configuration for Linux (WSL). To edit additional settings not shown here configuration and build. Use this edition is the complete files to allow easy debugging. This corresponds to CMAKeSettingsion.         Reference       Configuration for the used. "Debug" includes debug information in the compiled files to allow easy debugging. This corresponds to CMAKeSUID_TYPE.         RetWithDebindo       Context         Totestel       Context         Detect       Contochain file specify includes debug information. Any custom environment can be defined in CMakeSettingsjoon or you can select a proceed environment above. This setting maps to inheritEnvironment can be defined in CMakeSettingsjoon or you can select a proceed environment above. This setting maps to inheritEnvironment in CMakeSettingsjoon         Make totolchain file       Define totolchain file specify locations of compilers and toolchain utilities, and other target platorm and compiler related information. By default, Visual Studio will try and use the vcpkg toolchain file if this setting is unspecified.	na	
Select       Cancel         or configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings.json file. Add a new configuration to build and debug or exhibitions Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings.json.         Cancel         Cancel         Candidation name:         Ariflaration name:         Mendod have that identifies the configuration.         Mediane that identifies the configuration in the compiled files to allow easy debugging. This corresponds to construction.         Reverse         Mendod have that identifies the configuration. Any custom environment can be defined in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironments in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironment is in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironments in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironments in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironment is in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironment is in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironment is in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironment is in CMakeSettings.json or you can select a project devironment above. This setting as to inheritEnvironment is in CMakeSettings.json or you can select a project devironment above. This setting as to inheri	na	
Select       Cancel         to configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettingsjon file. Add a new configuration to build and debug or eVindows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettingsjon.         Extended       Edit JSON         Configuration name:       A friendly name that identifies the configuration.         x64:elease       Image: Status additional settings not shown here on the compiled files to allow easy debugging. This corresponds to CMAkeE, BullD, TYPE.         RetwinDeblind       Image: Status additional settings and the configuration.         x64:elease       Image: Status additional settings in the compiled files to allow easy debugging. This corresponds to cMAkeE, BullD, TYPE.         RetwinDeblind       Image: Status additional setting maps to inheritEnvironment can be defined in CMakeSettings joon or you can select a project files. Colorbain files specify locations of compilers and toolchain utilities, and other target plant advironment allower. This setting maps to inheritEnvironments in CMakeSettings joon.         max: x64_x64       Image: Telese         Image: Colorbain file       Image: Telese         Image: Colorbain file       Telese         Image: Colorbain file <td>na</td>	na	
be configure CMake project generation and build. Use this editor to edit settings defined in the underlying CMakeSettings json file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings json file. Add a new configuration to build and debug or e Windows Subsystem for Linux (WSL). To edit additional settings not shown here, go to CMakeSettings json. Edit JSON Ceneral Configuration name: A thendy name that Identifies the configuration. YetHease Configuration (Configuration) Retwindows Subsystem Configuration	na	
	bug ws (64-bit) with the MinGW development environment. MINGW_PREFIX must be set. (Debug) lease ws (64-bit) with the MinGW development environment. MINGW_PREFIX must be set. (RelWith ws (32-bit) with the Visual Studio development environment. (Debug) ws (32-bit) with the Visual Studio development environment. (RelWithDebInfo) ws (64-bit) with the Visual Studio development environment. (Debug) ws (64-bit) with the Visual Studio development environment. (RelWithDebInfo) ebug dows (32-bit) with the Visual Studio development environment. (Debug) elease dows (32-bit) with the Visual Studio development environment. (RelWithDebInfo)	

Save your new Cmake settings configuration and, before building the SDK, change the current configuration to the release created previously.

	x64-Debug
	x64-Debug
ł	x64-Release
	Manage Configurations

You can generate the cache and build the complete project (release) by right-clicking in CMakeLists.txt and selecting "Generate Cache" and then "Build."

x64-Release	Ŧ	🕨 Select Startup Item 👻 🛫		년 Live Share &
idarPy.cpp ect generation and b build and debug on o to CMakeSettingsj o to CMakeSettingsj src \out\build\x64	arPy.cpp CMakeSettingsjson ㅋ × 두 * t generation and build. Use this editor to edit settings defined in the underlying uild and debug on a remote Linux machine or the Windows Subsystem for Linux o CMakeSettingsjson.		<ul> <li>Solution Explorer - Folder View</li> <li>Search Solution Explorer - Folder View</li> <li>Search Solution Explorer - Folder View</li> <li>Search Solution Explorer - Folder View</li> <li>Leddar</li> <li>Leddar RoS</li> <li>LeddarRample</li> <li>LeddarRample</li> <li>LeddarTech</li> <li>LeddarTech</li> <li>Py_leddar_utils</li> <li>Telease</li> <li>Shared</li> <li>CMakeLists.txt</li> <li>CMakeSettings.json</li> </ul>	<ul> <li>4 ×</li> <li>w (Ctrl+ê)</li> <li>er\src)</li> <li>Add</li> <li>Open</li> <li>Open With</li> <li>Generate Cache</li> <li>CMake Cache (x64-Release Only)</li> <li>CMake Settings</li> <li>Configure Tasks</li> <li>CMate Generate Dependence</li> </ul>
	Solution Search D D D D D D D D D D D D D D D D D D D	n Explorer - Folder View Colution Explorer - Folder Solution Explorer - Folder Solution Explorer - Folder Colucion Explorer - Folder Colu	View (Ctrl+è) Add Open Open With Go To Git Changes Generate Cache for Ledo CMake Cache (x64-Debu CMake Settings for Ledo Set as Startup Item Configure Tasks Debug Build CMake Overview Pages Debug and Launch Settin Install	'd add a new in Visual Studio. darConfigurator4 ug Only) tarConfigurator4



Output	<b>→</b> ‡ ×
Show output from: Build	-   ≦_   ≦_ ≥   ஜ
[75/79] C:\PROGRA~2\MICROS~2\2019\COMMUN~1\VC\Tools [76/79] C:\PROGRA~2\MICROS~2\2019\COMMUN~1\VC\Tools	\MSVC\1424~1.263\D1N\HostX64\X64\c1.exe /nologo -DBUI
[77/79] cmd.exe /C "cd . && C:\PROGRA~2\MICROS~2\20	19\COMMUN~1\VC\Tools\MSVC\1424~1.283\bin\Hostx64\x64\1
[78/79] C:\PROGRA~2\MICROS~2\2019\COMMUN~1\VC\Tools	\MSVC\1424~1.283\bin\HostX64\x64\cl.exe /nologo /TP -
[79/79] cmd.exe /C "cd . && "C:\Program Files (x86)	<pre>\Microsoft Visual Studio\2019\Community\Common7\IDE\Co</pre>
Build succeeded.	-
	×
٩ 📖	P

The release can be found in .\LeddarSDK-master\src\out\build\x64-Release or in the folder selected in the configuration phase.

If you are using Python 3.9, you may get the following error:

😢 C4996	'PyEval_ThreadsInitialized': deprecated in 3.9	C:\Users\user464\Documents\Dev\SDK-2\LeddarSDK-master\src\o
😢 C4996	'PyEval_InitThreads': deprecated in 3.9	C:\Users\user464\Documents\Dev\SDK-2\LeddarSDK-master\src\o

You can correct the error by commenting the following lines in LeddarPy.cpp:



#### The release can be found in .\LeddarSDK-master\src\out\build\x64-Debug.

> opt > LeddarSDK-master > src > out > build > x64-Debug

Name	Date modified	Туре	Size
🖹 .cmake	2021-01-25 10:36 AM	File folder	
序 CMakeFiles	2021-01-25 10:45 AM	File folder	
阼 Testing	2021-01-25 10:36 AM	File folder	
📄 .ninja_deps	2021-01-25 10:48 AM	NINJA_DEPS File	18 KB
📄 .ninja_log	2021-01-25 10:48 AM	NINJA_LOG File	8 KB
📄 build.ninja	2021-01-25 10:45 AM	NINJA File	83 KB
📄 cmake_install.cmake	2021-01-25 10:36 AM	CMAKE File	2 KB
CMakeCache.txt	2021-01-25 10:36 AM	Text Document	16 KB
🏥 LC4.lib	2021-01-25 10:48 AM	Object File Library	30,047 KB
📑 LeddarExample.exe	2021-01-25 10:48 AM	Application	2,184 KB
🔓 LeddarExample.ilk	2021-01-25 10:48 AM	Incremental Linker File	12,342 KB
🖨 LeddarExample.pdb	2021-01-25 10:48 AM	Program Debug Data	6,068 KB
📄 rules.ninja	2021-01-25 10:45 AM	NINJA File	4 KB
VSInheritEnvironments.txt	2021-01-25 10:45 AM	Text Document	1 KB

### 1.1.1.1. Running Windows C++ Package

You can run LeddarExampleModule project by selecting CMakeLists.txt and starting the debug; otherwise, run the **LeddarExampleModule.exe** build after the compilation in the **release** folder.

.\LeddarSDK-master\src\out\build\x64-Debug\LeddarExample.exe



A simple example to connect and read data from the sensor is present in the SDK.

.\LeddarSDK\src\LeddarExample\LeddarExample.cpp

You can use the example as a reference for integration into your project.

#### Running a Leddar™ T16 sensor with C++

To run a Leddar T16 sensor through Ethernet connection, some small modifications to the code are required to visualize properly before compiling the SDK.

**Notes:** The SDK on the Leddar T16 allows basic communication with the sensor as echoes retrieval and configuration setting (GET and SET). However, the T16 Traffic version does not support the camera management nor the motor control (Pan and Tilt) options.

Due to the T16's frame rate, the data throughput cannot be viewed properly in the command prompt. Therefore, the detection values when using **LeddarExample.cpp** "seem" to remain constant and not to vary according to the environmental change, e.g., moving the sensor. This behavior is correct considering that the processing speed of an application layer is much higher than the visualization in the command prompt.

For a user-friendly visualization in the Windows command prompt, a few changes in **LeddarExample.cpp** are suggested below.

Add the lines highlighted in yellow and leave the rest of the code as is.

1. In the Function: displayEchoes, add a loop to show 1 out of 100 detections.

void displayEchoes( const std::string &aSensorName, LeddarDevice::LdSensor \*aSensor )

static int count = 0; LeddarDevice::LdSensorPixell \*IPix = dynamic\_cast<LeddarDevice::LdSensorPixell \*>( aSensor ); LeddarDevice::LdSensorDTec \*IDTec = dynamic\_cast<LeddarDevice::LdSensorDTec \*>( aSensor );

2. In the Callback Class: LdDisplayer, add a loop to show 1 out of 100 detections.

else if( aSender == mEchoes ) {
 //---- //T16 modification to refresh the command prompt in Windows
 static int T16\_count = 0;
 if (++T16\_count < 100)
 return;
 T16\_count = 0;
 //----- mEchoes->Lock( LeddarConnection::B\_GET );
std::cout << "Channel\tDistance\tAmplitude - Count = " << mEchoes->GetEchoCount() << " @ " << mEchoes-</pre>

std::vector<LeddarConnection::LdEcho> &lEchoes = \*( mEchoes->GetEchoes() );

3. In the Function: connectedMenu, modify the wait value, between displayEchoes request:

//-----//T16 modification to refresh the command promt in Windows LeddarUtils::LtTimeUtils::Wait( <mark>1</mark> ); //-----

>GetTimestamp() << std::endl;</pre>

© 2021 LeddarTech Inc. All rights reserved. This document may not be reproduced, distributed or disclosed, in whole or in part, without the express written consent of LeddarTech.



Running LeddarExample.cpp, it is possible to connect to the T16 sensor using any of the routines highlighted below.



### 1.1.2. Windows Python Package

**Notes:** The SDK in Windows is compatible only with Python versions > 3.8. Be sure you use the correct Python environment during compilation and building.

**ROS NOETIC for Windows uses Python 3.8.3.** Therefore, you need to compile LeddarPy in the same version of Python to work with Leddar in ROS for Windows.

If not done previously, download the latest version of Python from https://www.python.org/downloads/

Before building the Python SDK, add the **release** folder information to the setup.py.

By including to the library\_dirs the release folder and saving the file.

```
library dirs = ['../release','../out/build/x64-Release']
```

CMak	eSettings.json setup.py + ×
	•
	try:
	from setuptools import setup
	except ImportError:
	<pre>from distutils.core import setup</pre>
	from distutils.core import Extension
	import platform
	import numpy as np
	import sys
	import os
ę	<pre>extra_compile_args = ['-DBUILD_AUTO', '-DBUILD_CANBUS', '-DBUILD_CANBUS_KOMODO', '-DBUILD_DTEC', '-DBUILD_ETHERNET', '-DBUILD_M16', library_dirs = ['/release','/out/build/x64-Release'] include_dirs = ['./','/Leddar','/LeddarTech','/shared/','/shared/comm','/./libs/RapidJson', np.get_include()] libraries = ['LC4']</pre>
	<pre>if platform.system() != 'Windows':     libraries.append('usb-1.0')</pre>
	# define 'LEDDAR DEBUG' using e.g.
	# usan@machine.w/////wain/src/laddarDv& (evnort LENDAR DERLIG-1: nuthon satur nu installusan)
110 %	✓ No issues found

Python SDK uses NumPy libraries; therefore, they need to be installed.

To do so, you can open the Python environments and install NumPy using the command:

pip install numpy					
x64-Release    Current Document (setup.py) *   Python 3.9 (64-bit)				🖄 Live Share	Ŕ
1		<b>-</b> ☆	Python Environments		<b>4</b> ×
- etup	-	÷	Add Environment Python Software Foundation	E	
sion		BETT BELLER BELE	Python 3.9 (64-bit) Python Software Foundation	E	
			python27amd64 Custom environment	E	
		An	Packages (PyPI)		~
UTO', '-DBUILD_CANBUS', '-DBUILD_CANBUS_KOMODO', '-DBUILD_DTEC', '-DBUILD_ETHERNET', /out/build/x64-Release']	'-DBUILD_M16',		pip install numpy		×

To build the complete Python solution, run a Windows command prompt and navigate to:

.\LeddarSDK-master\src\LeddarPy

*Note:* Be sure you run a command prompt with the same x86 or x64 architecture as the previously compiled SDK. In our case, x64:

0F	
Bes	t match
CIN_	Command Prompt App X64
57-1	Visual C++ 2008 64-bit Command Prompt
5×**	Visual C++ 2008 64-bit Cross Tools Command Prompt
Sea	rch the web
Q	cmd - See web results
Q	cmd

Then run:

```
cd C:\opt\LeddarSDK-master\src\LeddarPy
```

py setup.py install --user

If you have multiple instances of Python installed, you can use the following to specify the exact version and architecture:

py -3.9-64 setup.py install --user

Note that the Python egg will be installed in a different path depending on the use of the –user command:

py setup.py install

"c:\users\user236\appdata\local\programs\python\python39\lib\site-packages\leddar-1.0-py3.9-win-amd64.egg"

#### OR:

LeddarTech

#### py setup.py install --user

"c:\users\user236\appdata\roaming\python\python39\site-packages\leddar-1.0-py3.9-win-amd64.egg"



If you get an error in the compilation, you might need to install NumPy libraries.

To do so, navigate to the Python environment in Visual Studio:

Тос	ls Extensions Window Help Sea	arch (Ctrl+	⊦Q)	م	Ledda	arPy		Sign in 오
	Get Tools and Features		Pyth	on 3.6 (64-bit)		Ë	⊟ _	
<b>*</b> 8	Connect to Database							
Ť	Connect to Server			••••••••••••••••••••••••••••••••••••••				
	Code Snippets Manager Ctrl+K, C	Ctrl+B	· ©	· 5 @ @ <b>/ -</b>				
	Choose Toolbox Items			trl+è)				
	NuGet Package Manager		'y' (3	of 3 projects)				
	Python		!	Issues and suggestions				
	Create GUID			Diagnostic Info				
	Error Lookup			Import coverage.py XM	IL file			
	Spy++		$\odot$	Cookiecutter Explorer				
	External Tools			Python Environments		(	Ctrl+K, Ctrl+ò	
	Command Line			Python Interactive Wind	low	,	Alt+I	
	Import and Export Settings			Python Debug Interactiv	ve		Shift+Alt+I	
	Customize		Varm					
₽	Options		Aarri	pie.py				

Then select the Python version you are using (in this case, Python 3.9 (64-bit) - Packages) and search for numpy.

### Then click Run command: pip install numpy.

Add <u>E</u> nvironment						
Python 2.7 (64-bit) Python Software Foundation	111	Python 3.9	(64-bit)			
Python 3.7 (64-bit)	-	Overview	numpy		3	×
Python Software Foundation		Packages (PyPI)	1.20.2)	• 1.20.3	×	^
Python 3.8 (64-bit) Python Software Foundation	<b>F1</b>		Run command: pip install numpy			
Python 3.9 (64-bit)			Install numpy-aarch64 (1.16.4)			
Python Software Foundation		J	Install numpy-alignments (0.0.2)			
			Install numpy-camera (0.0.2)			
			Install numpy-cloud (0.0.5)			
			Install numpy-datasets (0.0.2)			
			Install numpy-demo (1.23.0)			
			Install numpy-ext (0.9.3)			
			Install numpy-financial (1.0.0)			
			Install numpy-fracadf (0.4.0)			
			Install numpy-fracdiff (0.3.1)			

### 1.1.2.1. Running Windows Python Package

To run LeddarPyExample or LeddarPyAdvancedExample in Visual Studio 2019, you need to:

- 1. Modify LeddarPyExample or LeddarPyAdvancedExample connection parameters according to the sensor to use.
- 2. Set up the correct Python environment.

### 1.1.2.2. Modifying LeddarPyExample or LeddarPyAdvancedExample Connection Parameters

#### In .\LeddarSDK-master\src\LeddarPy\, you will find LeddarPyExample.py and LeddarPyAdvancedExample.py

By opening it with your favorite text editor or in Visual Studio, you will find in the code specification about the connection method.

Select the correct connection method according to your sensor and communication parameters.

### ##Use one of this connection method

### #Ethernet

# sensor\_list = leddar.get\_devices("Ethernet")
# print(leddar.device\_types["Ethernet"])
# dev.connect('192.168.0.2', leddar.device\_types["Ethernet"], 48630)
# connection = dev.connect('192.168.0.2', leddar.device\_types["Ethernet"], 48630)

#### #M16 Usb

```
# sensor_list = leddar.get_devices("Usb")
# dev.connect(sensor_list[0]['name'], leddar.device_types["Usb"])
```

```
##For any sensors with modbus serial communication (LeddarOne, Vu8 or M16)
# sensor_list = leddar.get_devices("Serial")
# dev.connect(sensor_list[0]['name'], leddar.device_types["Serial"])
```

```
##For Vu8/M16 sensors with CAN BUS communication
# dev.connect(Baud rate (kbps), Type of sensor, Tx ID (optionnal), Rx ID (optionnal))
```

```
# dev.connect('1000', leddar.device_types["M16Komodo"])
# dev.connect('1000', leddar.device_types["Vu8Komodo"])
```

### Running a Leddar Pixell with Python

**To run a Leddar Pixell through Ethernet connection**, comment the other connection methods, leaving only Ethernet enabled and change the IP parameter according to the sensor IP. The port must always be set at 48630.

LeddarTech®

#Ethernet
sensor\_list = leddar.get\_devices("Ethernet")
print(leddar.device\_types["Ethernet"])
connection = dev.connect('192.168.0.2', leddar.device\_types["Ethernet"], 48630)

You can run the example if the Python environment is properly set.



### Running a Leddar M16 through USB with Python

Tu run a Leddar M16 through USB connection, comment the other connection methods, leaving only M16 USB enabled.

```
#M16 Usb
sensor_list = leddar.get_devices("Usb")
dev.connect(sensor_list[0]['name'], leddar.device_types["Usb"])
```

You can run the example if the Python environment is properly set.

Command Prompt	_	$\times$
C:\opt\LeddarSDK-master\src\LeddarPy>python LeddarPyExample.py		^
resolution: 1 x 16, fov: 48.0		
Count:19		
timestamp:71735		
Channel - Distance - Amplitude - Flag - X - Y - Z - Timestamps		
0 - 2.1024628 - 515.8306 - 9 - 1.9424223 - 0.80457765 - 0.0 - 0		
1 - 2.002411 - 529.9684 - 9 - 1.8875556 - 0.66841847 - 0.0 - 0		
2 - 1.966095 - 538.0653 - 9 - 1.8851306 - 0.55840117 - 0.0 - 0		
3 - 1.9485321 - 539.8217 - 9 - 1.894694 - 0.4548758 - 0.0 - 0		
4 - 1.9226074 - 539.4639 - 9 - 1.8904132 - 0.35036737 - 0.0 - 0		
5 - 1.8835449 - 538.0569 - 9 - 1.8674309 - 0.24585195 - 0.0 - 0		
6 - 1.8999634 - 539.1158 - 9 - 1.8941064 - 0.14906941 - 0.0 - 0		
7 - 1.9021301 - 535.8698 - 9 - 1.9014783 - 0.049791962 - 0.0 - 0		
8 - 1.9139252 - 539.01044 - 9 - 1.91326930.05010072 - 0.0 - 0		
9 - 1.9105988 - 540.744 - 9 - 1.9047090.14990385 - 0.0 - 0		
10 - 1.8931732 - 540.46515 - 9 - 1.87697680.2471087 - 0.0 - 0		
11 - 1.5547943 - 569.2357 - 9 - 1.52875910.28333876 - 0.0 - 0		
12 - 1.6832123 - 276.03186 - 1 - 1.6367050.3929381 - 0.0 - 0		
13 - 0.9228058 - 101.77336 - 1 - 0.88480440.262091 - 0.0 - 0		
13 - 6.181183 - 7.8982697 - 64 - 5.926641.7555507 - 0.0 - 0		
14 - 0.9254303 - 39.32258 - 1 - 0.8723490.308915 - 0.0 - 0		
14 - 5.66626 - 7.388626 - 64 - 5.34125141.8914363 - 0.0 - 0		
15 - 0.83966064 - 23.688744 - 1 - 0.77574530.32132423 - 0.0 - 0		
15 - 5.530777 - 11.714348 - 64 - 5.10977172.1165366 - 0.0 - 0		
C:\opt\LeddarSDK-master\src\LeddarPy>		

### Running a Leddar M16, Vu8, or LeddarOne Modbus with Python

To run a LeddarOne, Vu8 or M16 sensor with Modbus connection, comment the other connection methods, leaving only Modbus serial communication enabled, as it is possible to have multiple COM ports available. To select the correct one, try to modify the code as follows:

```
##For any sensors with modbus serial communication (LeddarOne, Vu8 or M16)
sensor_list = leddar.get_devices("Serial")
for port in sensor_list:
    print("COM ports available: ", port)
dev.connect(sensor_list[1]['name'], leddar.device_types["Serial"])
```

The item highlighted in yellow must correspond to the COM the sensor is connected to. In our case:

sensor\_list[0]  $\rightarrow$  Point to COM3 sensor\_list[1]  $\rightarrow$  Point to COM11 (where the sensor is connected)

If the Python environment is properly set, you can run the example.

Command Prompt	-		×
			^
C:\opt\LeddarSDK-master\src\LeddarPy>python LeddarPyExample.py			
COM ports avaliable: {'name': '\\\\.\\COM3', 'type': 2, 'address': '\	.///.//C	OM3'}	
COM ports avaliable: {'name': '\\\\.\\COM11', 'type': 2, 'address': '	////.//	COM11'}	
resolution: 1 x 8, fov: 45.0			
Count:7			
timestamp:1952365			
Channel - Distance - Amplitude - Flag - X - Y - Z - Timestamps			
7 - 2.1678467 - 121.07143 - 1 - 2.04112320.7303256 - 0.0 - 0			
6 - 2.0451355 - 171.16179 - 1 - 1.98384540.49692738 - 0.0 - 0			
5 - 2.0409393 - 194.2578 - 1 - 2.01884910.299468 - 0.0 - 0			
4 - 2.0401459 - 200.94426 - 1 - 2.03768850.10010521 - 0.0 - 0			
3 - 2.0255127 - 187.65327 - 1 - 2.023073 - 0.0993872 - 0.0 - 0			
2 - 2.0070343 - 194.06447 - 1 - 1.9853112 - 0.2944931 - 0.0 - 0			
1 - 1.9000702 - 28.018726 - 64 - 1.8431275 - 0.4616794 - 0.0 - 0			
C:\opt\LeddarSDK-master\src\LeddarPy>			~

### Running a Leddar T16 with Python

**To run a Leddar T16 sensor through Ethernet connection**, comment the other connection methods, leaving only Ethernet enabled and change the IP parameter according to the sensor IP. The port must always be set at 48620.

### #Ethernet

sensor\_list = leddar.get\_devices("Ethernet")
print(leddar.device\_types["Ethernet"])
dev.connect('192.168.0.2', leddar.device\_types["Ethernet"], 48620)
connection = dev.connect('192.168.0.2', leddar.device\_types["Ethernet"], 48620)

You can run the example if the Python environment is properly set:

LeddarPyExample.py

Command Prompt	_	×
C:\opt\LeddarSDK-master\src\LeddarPy>python LeddarPyExample.py		^
65536		
resolution: 1 x 16, fov: 36.4		
Count:16		
timestamp:31064		
Channel - Distance - Amplitude - Flag - X - Y - Z - Timestamps		
0 - 1.6959229 - 125.525635 - 9 - 1.6212772 - 0.49760872 - 0.0 - 0		
1 - 1.3292389 - 165.00491 - 9 - 1.2852134 - 0.33926806 - 0.0 - 0		
2 - 1.3028107 - 178.16006 - 9 - 1.2718673 - 0.28225735 - 0.0 - 0		
3 - 1.2259521 - 174.04199 - 9 - 1.2064344 - 0.21788709 - 0.0 - 0		
4 - 1.18/088 - 126.5/1045 - 9 - 1.1/56432 - 0.1644413 - 0.0 - 0		
5 - 1.1983643 - 126.19562 - 9 - 1.192465 - 0.11876109 - 0.0 - 0		
0 - 1.2003032 - 120.280150 - 9 - 1.1982348 - 0.0/1450000 - 0.0 - 0		
7 - 1.2214355 - 120.1/817 - 9 - 1.2211949 - 0.024247715 - 0.0 - 0		
0 = 1.1994019 = 120.29497 = 9 = 1.1991059 = -0.023010507 = 0.0 = 0 0 = 1.2030207 = 124.68103 = 0 = 1.2017525 = -0.07285207 = 0.0 = 0		
10 - 1 $3/10399 - 125 6788 - 9 - 1 337/231 - 0 13319793 - 0.0 - 0$		
10 - 1.5440555 - 125.0700 - 5 - 1.5574254 - 0.15515755 - 0.0 - 0		
12 - 1.457962 - 141.78345 - 9 - 1.43475060.25912192 - 0.0 - 0		
13 - 1.4758148 - 175.17027 - 9 - 1.44076240.31973913 - 0.0 - 0		
14 - 1.5309296 - 155.92041 - 9 - 1.48022380.39074656 - 0.0 - 0		
15 - 1.670517 - 125.60435 - 9 - 1.59698960.49015424 - 0.0 - 0		
		$\sim$



**Notes:** The SDK on the Leddar T16 allows basic communication with the sensor as echoes retrieval and configuration setting (GET and SET). However, the T16 Traffic version does neither support the camera management nor the motor control (Pan and Tilt) options.

Due to the T16's frame rate, the data throughput cannot be viewed properly in the command prompt. Therefore, the detection values when using **LeddarPyAdvancedExample.py** "seem" to remain constant and not to vary according to the environmental change, e.g., moving the sensor. This behavior is correct as the processing speed of an application layer is much higher than the visualization in the command prompt.

For a user-friendly visualization in the Windows command prompt, a few changes in **LeddarPyAdvancedExample.py** are suggested below.

**1.** Add a global variable and implement a loop to show 1 out of 100 detections.

import leddar import time

<mark>global</mark> T16\_count T16\_count = 0

leddar.enable\_debug\_trace(True)

```
#Callback functions for the data thread
def echoes_callback(echoes):
    data = echoes["data"]
```

```
#To avoid display all the ECHOES
global T16_count
if (T16_count < 100) :
    #print("Count:" + str(T16_count))
    T16_count = T16_count + 1
    return
T16_count = 0</pre>
```

```
#To avoid display of too much lines
increment = 1
if len(data) > 100 :
    increment = 100
```

print("Count:" + str(len(data)))

2. Modify the delay between two requests to be a small value:

#Optional : set the delay between two request to the sensor dev.set\_data\_thread\_delay(1)

You can find additional information in section "1.1.2.2 Modifying LeddarPyExample or LeddarPyAdvancedExample Connection Parameters."

### 2. Installing ROS for Windows

Go to https://wiki.ros.org/Installation/Windows.

Follow the required steps.

```
In step 3, remember to include "Desktop development with C++" workload.
```

R	Visual Studio Community 2019	Pause
	Downloaded	
	Installing: package 238 of 305 33% Microsoft.VisualCpp.Redist.14.Latest	
	✓ Start after installation	
	Release notes	

Steps 5.2 and 6.1 are optional and are not required for working with Leddar sensors.

After completing step 7, close and reopen the ROS command prompt.

You are now ready to start using ROS.

You can check this by opening as Administrator (step 6) and running:

roscore

If you get errors regarding any missing dll, reinstall ROS by running in the ROS command prompt:

```
choco install ros-noetic-desktop_full -y --execution-timeout=0 --force
```

### Then relaunch:

roscore



### 3. Running Leddar Sensors in ROS for Windows

You will find the ROS driver for Leddar sensors in LeddarSDK-master. Navigate to .\LeddarSDK-master\src\Leddar\_Ros

*Note:* ROS NOETIC for Windows uses Python 3.8.3. Therefore, you need to compile LeddarPy in the same version of Python to work with Leddar sensors in ROS for Windows.

### 3.1. Installing LeddarPy and NumPy in ROS

Note: It is mandatory to update LeddarPy as follows to use the latest Leddar\_ROS driver.

### Using the ROS command prompt

Update NumPy by running:

py -m pip install --upgrade numpy
py -3.8 -m pip install --upgrade numpy

Then, check where Python is installed by running:

where python python --version

Python in the Windows directory and ROS related directory will appear:

::\opt\catkin\_ws>python --version Python 3.8.3

c:\opt\catkin\_ws>where python C:\opt\ros\noetic\x64\python.exe

Typically:

C:\opt\ros\**noetic**\x64\python.exe

### Using the ROS command prompt

### 3.1.1. Building LeddarPy

Navigate to the LeddarPy folder by running:

cd C:\opt\LeddarSDK-master\src\LeddarPy

Build it in Python 3.8.3 by running:

py -3.8-64 setup.py build

The build library will be created in "LeddarSDK-master\src\LeddarPy\build\lib.win-amd64-3.8"

Copy the **leddar.cp38-win\_amd64.pyd** file into C:\opt\ros\noetic\x64.

cd C:\opt\LeddarSDK-master\src\LeddarPy
copy build\lib.win-amd64-3.8\leddar.cp38-win\_amd64.pyd C:\opt\ros\noetic\x64

### 3.2. Preparing the catkin Workspace

Using File Explorer, create a new folder under c:\opt\

catkin\_ws

Create a sub-folder inside catkin\_ws

src

Alternatively, using the ROS command prompt, run:

```
cd c:\opt\
mkdir catkin_ws
cd catkin_ws
mkdir src
```

Open the ROS command prompt and run:

cd c:\opt\catkin\_ws\src
catkin\_init\_workspace

With Windows Explorer, copy from any other catkin workspace into the src folder:

```
leddar_ros
ros_numpy
                 📜 🛛 🖌 📘 🛨 🛛 src
                           Home
                                    Share
                                              View
                                     > This PC > Windows (C:) > opt > catkin ws > src
                 ←
                              \mathbf{\Lambda}
                                  Date modified
                  Name
                                                                                    Туре
                   📕 leddar_ros
                                                               2020-01-30 09:42...
                                                                                    File folder
                   ros_numpy
                                                               2020-01-30 09:42... File folder
```

### Using the ROS command prompt, run:

cd c:\opt\catkin\_ws
catkin\_make



Close and re-open your ROS command prompt.

Update the catkin environment by running:

```
c:\opt\catkin_ws\devel\setup.bat
cd c:\opt\catkin_ws
catkin_make
```



### 3.3. Running Leddar Pixell With ROS for Windows

Update the catkin environment using the ROS command prompt:

```
c:\opt\catkin_ws\devel\setup.bat
cd c:\opt\catkin_ws
catkin_make
```

Launch Pixell example: To avoid modifying the .launch file, it is possible to send all parameters directly in the command line:

roslaunch leddar\_ros example.launch param1:=192.168.0.2 device\_type:=Ethernet param3:=48630
param4:=0

Alternatively, it is possible to modify the example.launch to reflect the required parameters:

```
<arg name="param1" default="192.168.0.2" />
<arg name="device_type" default="Ethernet" />
<arg name="param3" default="48630"/>
<arg name="param4" default="0"/>
```

Then run the launch file:

roslaunch leddar\_ros example.launch





Below is a diagram of the representation in Rviz according to the sensor orientation:



It is possible to move the sensor axes positions with regards to the map axes position using **device.xml**. See the sections highlighted in blue below.

### 3.3.1. Leddar Pixell ROS Code Examples

"leddar\_ros\launch\example.launch"

```
<launch>
 <!-- Connection info, see python leddar.Device.connect for more info -->
 <arg name="param1" default="192.168.0.2" />
 <!-- [All] value returned by GetDeviceList['name'],
  [Serial] serial port com,
  [USB] serial number,
  [SPI-FTDI] FTDI cable id (use get_devices("SpiFTDI")),
  [CANBus komodo] Baudrate in kbps,
  [Ethernet] Ip address -->
 <arg name="device_type" default="Ethernet" />
 <!-- (optional but recommended) Device type | connection type -
  Both are mandatory for CANBus protocol (from leddar.device_types dictionnary) -->
 <arg name="param3" default="48630"/>
 <!-- (optional) [Serial] modbus address (default 1),
  [CANBus komodo] Tx (default 0x750),
    [Ethernet] port (default 48630) -->
  <arg name="param4" default="0"/>
  <!-- param4: (optional) [Serial] baudrate (default 115200),
    [CANBus komodo] Rx (default 0x740),
    [Ethernet] Communication timeout -->
  <include file="$(find leddar ros)/config/device.xml" ns="LeddarTech 1" pass all args="true">
  </include>
  <node pkg="rviz" type="rviz" name="rviz" args="-d $(find leddar ros)/rviz/example.rviz"</pre>
output="screen"/>
</launch>
```

### "leddar\_ros\config\**device.xml**"

```
<launch>
 <!-- Connection info, see python leddar.Device.connect for more info -->
<arg name="param1" default="192.168.0.2" />
 <!-- [All] value returned by GetDeviceList['name'],
 [Serial] serial port com,
 [USB] serial number,
 [SPI-FTDI] FTDI cable id (use get_devices("SpiFTDI")),
 [CANBus komodo] Baudrate in kbps.
 [Ethernet] Ip address -->
 <arg name="device_type" default="Ethernet" />
 <!-- (optional but recommended) Device type | connection type -
  Both are mandatory for CANBus protocol (from leddar.device_types dictionnary) -->
<arg name="param3" default="48630"/>
 <!-- (optional) [Serial] modbus address (default 1),
 [CANBus komodo] Tx (default 0x750),
 [Ethernet] port (default 48630) -->
<arg name="param4" default="0"/>
 <!-- param4: (optional) [Serial] baudrate (default 115200),
 [CANBus komodo] Rx (default 0x740),
 [Ethernet] Communication timeout -->
 <arg name="frame_id" default="sensor_frame" />
 <arg name="parent_frame_id" default="map" />
<!-- Position of the Sensor <arg name="position" default="X Y Height" /> -->
 <arg name="position" default="0 0 0" />
 <!-- Euler_angles in ROS are (yaw, pitch, roll) in rad
 rotate about fixed X-axis by Roll
 rotate about fixed Y-axis by Pitch
 rotate about fixed Z-axis by Yaw
 <arg name="rotation" default="0 0 0" />
 <!-- Quaternion -->
 <!--<arg name="rotation" default="-0.707106781 0 0 0.707106781" /> -->
 <node pkg="leddar ros" type="device.py" name="sensor">
  <param name="device_type" value="$(arg device_type)" />
  <param name="frame_id" value="$(arg frame_id)" />
  <param name="param1" value="$(arg param1)" />
  <param name="param3" value="$(arg param3)" />
  <param name="param4" value="$(arg param4)" />
 </node>
<node pkg="tf" type="static_transform_publisher" name="leddar_1_to_map"
   args="$(arg position) $(arg rotation) $(arg parent_frame_id) $(arg frame_id) 10" />
</launch>
```

### 3.4. Running Leddar Vu8 With ROS for Windows

Update the catkin environment using the ROS command prompt:

```
c:\opt\catkin_ws\devel\setup.bat
cd c:\opt\catkin_ws
catkin_make
```

Launch Vu8 example: To avoid modifying the .launch file, it is possible to send all parameters in the command line:

```
roslaunch leddar_ros example.launch param1:=\\.\COM11 device_type:=Serial param3:=1
param4:=0
```

Alternatively, it is possible to modify the example.launch to reflect the required parameters:

```
<arg name="param1" default="\\.\COM11"/>
<arg name="device_type" default="Serial" />
<arg name="param3" default="1"/>
<arg name="param4" default="0"/>
```

### Then run the launch file:

### roslaunch leddar\_ros example.launch





Below is a diagram of the representation in Rviz according to the sensor orientation:



It is possible to move the sensor axes positions with regards to the map axes position using **device.xml**. See the sections highlighted in blue below.

### 3.4.1. Leddar Vu8 ROS Code Examples

"leddar\_ros\launch\example.launch"

<launch> <!-- Connection info, see python leddar.Device.connect for more info --> <arg name="param1" default="\\.\COM11" /> <!-- [All] value returned by GetDeviceList['name'], [Serial] serial port com, [USB] serial number, [SPI-FTDI] FTDI cable id (use get\_devices("SpiFTDI")), [CANBus komodo] Baudrate in kbps, [Ethernet] Ip address --> <arg name="device\_type" default="Serial" /> <!-- (optional but recommended) Device type | connection type -Both are mandatory for CANBus protocol (from leddar.device\_types dictionnary) --> <arg name="param3" default="1"/> <!-- (optional) [Serial] modbus address (default 1), [CANBus komodo] Tx (default 0x750), [Ethernet] port (default 48630) --> <arg name="param4" default="0"/> <!-- param4: (optional) [Serial] baudrate (default 115200), [CANBus komodo] Rx (default 0x740), [Ethernet] Communication timeout -->

<include file="\$(find leddar\_ros)/config/device.xml" ns="LeddarTech\_1" pass\_all\_args="true"> </include>

<node pkg="rviz" type="rviz" name="rviz" args="-d \$(find leddar\_ros)/rviz/example.rviz" output="screen"/> </launch>

### "leddar\_ros\config\**device.xml**"

<launch> <!-- Connection info, see python leddar.Device.connect for more info --> <arg name="param1" default="\\.\COM11" /> <!-- [All] value returned by GetDeviceList['name'], [Serial] serial port com, [USB] serial number, [SPI-FTDI] FTDI cable id (use get\_devices("SpiFTDI")), [CANBus komodo] Baudrate in kbps, [Ethernet] Ip address --> <arg name="device\_type" default="Serial" /> <!-- (optional but recommended) Device type | connection type -Both are mandatory for CANBus protocol (from leddar.device types dictionnary) --> <arg name="param3" default="1"/> <!-- (optional) [Serial] modbus address (default 1), [CANBus komodo] Tx (default 0x750), [Ethernet] port (default 48630) --> <arg name="param4" default="0"/>

```
<!-- param4: (optional) [Serial] baudrate (default 115200),
[CANBus komodo] Rx (default 0x740),
[Ethernet] Communication timeout -->
```

```
<arg name="frame_id" default="sensor_frame" />
<arg name="parent_frame_id" default="map" />
<!-- Position of the Sensor <arg name="position" default="X Y Height" /> -->
<arg name="position" default="0 0 0" />
<!-- Euler_angles in ROS are (yaw, pitch, roll) in rad
rotate about fixed X-axis by Roll
rotate about fixed Y-axis by Pitch
rotate about fixed Z-axis by Yaw
--->
<arg name="rotation" default="0 0 0" />
<!-- Quaternion -->
<!--<arg name="rotation" default="-0.707106781 0 0 0.707106781" /> -->
<node pkg="leddar_ros" type="device.py" name="sensor">
<param name="device type" value="$(arg device type)" />
```

```
<param name="frame_id" value="$(arg frame_id)" />
<param name="param1" value="$(arg param1)" />
<param name="param3" value="$(arg param3)" />
<param name="param4" value="$(arg param4)" />
</node>
```

```
<node pkg="tf" type="static_transform_publisher" name="leddar_1_to_map"
args="$(arg position) $(arg rotation) $(arg parent_frame_id) $(arg frame_id) 10" />
```

</launch>

### 3.5. Running Leddar M16 With ROS for Windows

Use the ROS command prompt update catkin environment:

c:\opt\catkin\_ws\devel\setup.bat
cd c:\opt\catkin\_ws
catkin\_make

Launch M16 (connected through USB) example: To avoid modifying the .launch file, it is possible to send all parameters directly in the command line:

roslaunch leddar\_ros example.launch param1:=AK47027 device\_type:=Usb param3:=1 param4:=0

*Note:* AK47027 is the unit's serial number.

For a Leddar IS16 connected through USB, the only change is the SN:

roslaunch leddar\_ros example.launch param1:=<mark>AM05002</mark> device\_type:=Usb param3:=1 param4:=0

Alternatively, it is possible to modify the example.launch to reflect the required parameters:

<arg name="param1" default="AK47027"/> <arg name="device\_type" default="Usb" />

```
<arg name="param3" default="1"/> <arg name="param4" default="0"/>
```

### Then run the launch file:

roslaunch leddar\_ros example.launch



Below is a diagram of the representation in Rviz according to the sensor orientation:







It is possible to move the sensor axes positions with regards to the map axes position using **device.xml**. See the sections highlighted in blue below.

### 3.5.1. Leddar M16/IS16 ROS Code Examples

"leddar\_ros\launch\example.launch"

<launch> <!-- Connection info, see python leddar.Device.connect for more info --> <arg name="param1" default="AK47027" /> <!-- [All] value returned by GetDeviceList['name'], [Serial] serial port com, [USB] serial number, [SPI-FTDI] FTDI cable id (use get\_devices("SpiFTDI")), [CANBus komodo] Baudrate in kbps, [Ethernet] Ip address --> <arg name="device type" default="Usb" /> <!-- (optional but recommended) Device type | connection type -Both are mandatory for CANBus protocol (from leddar.device\_types dictionnary) --> <arg name="param3" default="1"/> <!-- (optional) [Serial] modbus address (default 1), [CANBus komodo] Tx (default 0x750), [Ethernet] port (default 48630) --> <arg name="param4" default="0"/> <!-- param4: (optional) [Serial] baudrate (default 115200),</pre> [CANBus komodo] Rx (default 0x740), [Ethernet] Communication timeout --> <include file="\$(find leddar ros)/config/device.xml" ns="LeddarTech 1" pass all args="true">

</include>

<node pkg="rviz" type="rviz" name="rviz" args="-d \$(find leddar\_ros)/rviz/example.rviz" output="screen"/> </launch>

### "leddar\_ros\config\device.xml"

```
<launch>
 <!-- Connection info, see python leddar.Device.connect for more info -->
 <arg name="param1" default="AK47027" />
 <!-- [All] value returned by GetDeviceList['name'],
 [Serial] serial port com,
 [USB] serial number,
 [SPI-FTDI] FTDI cable id (use get_devices("SpiFTDI")),
 [CANBus komodo] Baudrate in kbps,
 [Ethernet] Ip address -->
 <arg name="device_type" default="Usb" />
 <!-- (optional but recommended) Device type | connection type -
 Both are mandatory for CANBus protocol (from leddar.device_types dictionnary) -->
 <arg name="param3" default="1"/>
 <!-- (optional) [Serial] modbus address (default 1),
 [CANBus komodo] Tx (default 0x750),
 [Ethernet] port (default 48630) -->
 <arg name="param4" default="0"/>
 <!-- param4: (optional) [Serial] baudrate (default 115200),
 [CANBus komodo] Rx (default 0x740),
 [Ethernet] Communication timeout -->
 <arg name="frame_id" default="sensor_frame" />
 <arg name="parent frame id" default="map" />
<!-- Position of the Sensor <arg name="position" default="X Y Height" /> -->
 <arg name="position" default="0 0 0" />
 <!-- Euler_angles in ROS are (yaw, pitch, roll) in rad
 rotate about fixed X-axis by Roll
 rotate about fixed Y-axis by Pitch
 rotate about fixed Z-axis by Yaw
 -->
 <arg name="rotation" default="0 0 0" />
 <!-- Ouaternion -->
 <!--<arg name="rotation" default="-0.707106781 0 0 0.707106781" /> -->
 <node pkg="leddar ros" type="device.py" name="sensor">
  <param name="device_type" value="$(arg device_type)" />
  <param name="frame_id" value="$(arg frame_id)" />
  <param name="param1" value="$(arg param1)" />
  <param name="param3" value="$(arg param3)" />
  <param name="param4" value="$(arg param4)" />
 </node>
 <node pkg="tf" type="static transform publisher" name="leddar 1 to map"
   args="$(arg position) $(arg rotation) $(arg parent_frame_id) $(arg frame_id) 10" />
```

### </launch>

### 3.6. Running Leddar T16 With ROS for Windows

Use the ROS command prompt update catkin environment:

```
c:\opt\catkin_ws\devel\setup.bat
cd c:\opt\catkin_ws
catkin_make
```

Launch T16 example: To avoid modifying the .launch file, it is possible to send all parameters directly in the command line:

roslaunch leddar\_ros example.launch param1:=192.168.0.2 device\_type:=Ethernet param3:=48620
param4:=0

Alternatively, it is possible to modify the example.launch to reflect the required parameters:

```
<arg name="param1" default="192.168.0.2" />
<arg name="device_type" default="Ethernet" />
<arg name="param3" default="48620"/>
<arg name="param4" default="0"/>
```

Then run the launch file:

roslaunch leddar\_ros example.launch





Below is a diagram of the representation in Rviz according to the sensor orientation:



It is possible to move the sensor axes positions with regards to the map axes position using **device.xml**. See the sections highlighted in blue below.

### 3.6.1. Leddar T16 ROS Code Examples

"leddar\_ros\launch\example.launch"

### <launch> <!-- Connection info, see python leddar.Device.connect for more info --> <arg name="param1" default="192.168.0.2" /> <!-- [All] value returned by GetDeviceList['name'], [Serial] serial port com, [USB] serial number, [SPI-FTDI] FTDI cable id (use get\_devices("SpiFTDI")), [CANBus komodo] Baudrate in kbps, [Ethernet] Ip address --> <arg name="device\_type" default="Ethernet" /> <!-- (optional but recommended) Device type | connection type -</p> Both are mandatory for CANBus protocol (from leddar.device types dictionnary) --> <arg name="param3" default="48620"/> <!-- (optional) [Serial] modbus address (default 1), [CANBus komodo] Tx (default 0x750), [Ethernet] port (default 48630) --> <arg name="param4" default="0"/> <!-- param4: (optional) [Serial] baudrate (default 115200), [CANBus komodo] Rx (default 0x740), [Ethernet] Communication timeout -->

<include file="\$(find leddar\_ros)/config/device.xml" ns="LeddarTech\_1" pass\_all\_args="true"> </include>

<node pkg="rviz" type="rviz" name="rviz" args="-d \$(find leddar\_ros)/rviz/example.rviz" output="screen"/> </launch>

"leddar\_ros\config\device.xml"

<launch>

```
<!-- Connection info, see python leddar.Device.connect for more info -->
<arg name="param1" default="192.168.0.2" />
<!-- [All] value returned by GetDeviceList['name'],
[Serial] serial port com,
[USB] serial number,
[SPI-FTDI] FTDI cable id (use get_devices("SpiFTDI")),
[CANBus komodo] Baudrate in kbps,
```

[Ethernet] Ip address -->

<arg name="device\_type" default="Ethernet" />

<!-- (optional but recommended) Device type | connection type -

Both are mandatory for CANBus protocol (from leddar.device\_types dictionnary) -->

<arg name="param3" default="48620"/>

<!-- (optional) [Serial] modbus address (default 1),

[CANBus komodo] Tx (default 0x750),

[Ethernet] port (default 48630) -->

<arg name="param4" default="0"/>

<!-- param4: (optional) [Serial] baudrate (default 115200),

[CANBus komodo] Rx (default 0x740),

[Ethernet] Communication timeout -->

<arg name="frame\_id" default="sensor\_frame" /> <arg name="parent\_frame\_id" default="map" />

```
<!-- Position of the Sensor <arg name="position" default="X Y Height" /> -->
<arg name="position" default="0 0 0" />
<!-- Euler_angles in ROS are (yaw, pitch, roll) in rad
rotate about fixed X-axis by Roll
rotate about fixed Y-axis by Pitch
rotate about fixed Z-axis by Yaw
```

-->

<arg name="rotation" default="0 0 0" />

<!-- Quaternion --> <!--<arg name="rotation" default="-0.707106781 0 0 0.707106781" /> -->

<node pkg="leddar\_ros" type="device.py" name="sensor"> <param name="device\_type" value="\$(arg device\_type)" /> <param name="frame\_id" value="\$(arg frame\_id)" /> <param name="param1" value="\$(arg param1)" /> <param name="param3" value="\$(arg param3)" /> <param name="param4" value="\$(arg param4)" /> </node>

```
<node pkg="tf" type="static_transform_publisher" name="leddar_1_to_map"
args="$(arg position) $(arg rotation) $(arg parent_frame_id) $(arg frame_id) 10" />
```

</launch>

LeddarTech® has made every effort to ensure that the information contained in this document is accurate. Any information herein is provided "AS IS." LeddarTech shall not be liable for any errors or omissions herein or for any damages arising out of or related to the information provided in this document. LeddarTech reserves the right to modify design, characteristics and products at any time, without notice, at its sole discretion.

LeddarTech does not control the installation and use of its products and shall have no liability if a product is used for an application for which it is not suited. You are solely responsible for (1) selecting the appropriate products for your application, (2) validating, designing and testing your application and (3) ensuring that your application meets applicable safety and security standards.

Furthermore, LeddarTech products are provided only subject to LeddarTech's Sales Terms and Conditions or other applicable terms agreed to in writing. By purchasing a LeddarTech product, you also accept to carefully read and to be bound by the information contained in the User Guide accompanying the product purchased.

Leddar, LeddarTech, LeddarEngine, LeddarVision, LeddarSP, LeddarCore, VAYADrive, VayaVision and related logos are trademarks or registered trademarks of LeddarTech Inc. and its subsidiaries. All other brands, product names and marks are or may be trademarks or registered trademarks used to identify products or services of their respective owners.

#### About LeddarTech

LeddarTech is a leader in environmental sensing platforms for autonomous vehicles and advanced driver assistance systems. Founded in 2007, LeddarTech has evolved to become a comprehensive end-to-end environmental sensing company by enabling customers to solve critical sensing and perception challenges across the entire value chain of the automotive and mobility market segments. With its LeddarVision™ sensorfusion and perception platform and its cost-effective, scalable and versatile LiDAR development solution for automotive-grade solid-state LiDARs based on the LeddarEngine™, LeddarTech enables Tier 1-2 automotive system integrators to develop full-stack sensing solutions for autonomy level 1 to 5. These solutions are actively deployed in autonomous shuttles, trucks, buses, delivery vehicles, smart cities/factories and robotaxi applications. The company is responsible for several innovations in cutting-edge automotive and mobility remote-sensing applications, with over 100 patented technologies (granted or pending) enhancing ADAS and autonomous driving capabilities.

For more information: sales@leddartech.com





CANADA - USA - AUSTRIA - FRANCE - GERMANY - ITALY - ISRAEL - HONG KONG - CHINA

Head Office

4535, boulevard Wilfrid-Hamel, Suite 240 Québec (Québec) G1P 2J7, Canada Ieddartech.com

Phone: + 1-418-653-9000 Toll-free: 1-855-865-9900