NaoLab Overview

RobotHandler

The **RobotHandler project** provides a list of interfaces that can be used to connect your work to the Nao robot from a remote computer. Developped in C++, the interfaces, running as drivers on a remote computer, dialog with the Nao robot modules through the network. By using the **DataProxy library**, the user is able to reach robot's data directly in his application and send back commands to the robot. Developped in C++, the **DataProxy** library can be easily interface with any programming langage, like Matlab/C++.

TOC

Tree Folder Overview

CMake_Modules

Contains a cmake file used by the compilation process to locate the DataProxy library.

CommonFiles

Contains header file that defines common structures used by the DataProxy library and the NaoRemoteDriver applications.

DataProxy

Contains C++ files that are used to compile the DataProxy library.

HeadPoseDemo

Contains C++ files that are used to compile a small example of live video processing application in C++.

MatlabInterface

Contains C++ files that are used to compile matlab mexa files using mex.

MatlabSoundLocalizationDemo

Contains matlab files that are used to run a simple example of speaker localization.

NaoAVRecorder

Contains C++ files that are used to compile an application that can be used to record Audio and Video.

NaoRemoteDriver

Contains C++ files that are used to compile the different drivers which remotely dialog with the Nao robot.

Compilation Manual

Warning : *Modifying the TreeFolder can cause compilation errors.*

Brief

1. Compile the NaoRemoteDriver
2. Compile the DataProxy library
3. Compile examples

NaoRemoteDriver

> ***Requirement :*** *naoqi-sdk-2.1.2.17-linux64 installed on the computer*

The **NaoRemoteDriver** folder contains a global **CMakeLists.txt** that compiles all the drivers.

Compilation steps :

- Open a terminal in **RobotHandler/NaoRemoteDriver/** folder
- Then type in terminal : `mkdir Build` `cd Build` `cmake .. -DNAOQI_SDK_DIRECTORY=/PATH_TO/naoqi-sdk-2.1.2.17-linux64/` `make`

Drivers executables are in the **RobotHandler/NaoRemoteDriver/Build/bin/** folder.

DataProxy

> ***Requirement :*** *Boost library installed on the computer (generally in **/usr/lib/** and **/usr/include**)*

The DataProxy folder contains a **CMakeLists.txt** that compiles and installs the **DataProxy** library.

Compilation steps :

- Open a terminal in **RobotHandler/DataProxy/** folder
- Then type in the terminal : `mkdir Build` `cd Build` `cmake .. #installation path is set by default to : /usr/local/` `cmake .. -DDATA_PROXIES_INSTALL_DIRECTORY=/PATH_TO_INSTALL/ # to fix the installation path` `make install #` `you may need to be root for the installation to be done without error`

Note :

- if you precise : **-DDATA_PROXIES_INSTALL_DIRECTORY=/PATH_TO_INSTALL/**
  - **libdataproxies.so** is to be installed in **/PATH_TO_INSTALL/lib/**
  - **dataproxies.h** is to be installed in **/PATH_TO_INSTALL/include/**
- if you do not precise any path
  - **libdataproxies.so** is to be installed in **/usr/local/lib**
  - **dataproxies.h** in **/usr/local/include**
- to uninstall these files, delete : **libdataproxies.so** and **dataproxies.h**

HeadPoseDemo

> ***Requirement :*** *OpenCV library installed on the computer*

The **HeadPoseDemo** folder contains **CMakeLists.txt** that compiles this C++ example.

Compilation steps :

- Open a terminal in **RobotHandler/HeadPoseDemo/** folder
- Then type in the terminal : `mkdir Build` `cd Build` `cmake .. #` if **DataProxy** library has been installed in the default location `/usr/local/` `cmake .. -DDATAPROXIES_SDK_DIRECTORY=/PATH_TO_DATAPROXY_SDK/ #` if **DataProxy** library has been installed in other location `make`

The executable **naoHeadPoseDemoRelWithDebInfo** is in the **RobotHandler/HeadPoseDemo/Build/** folder.

MatlabInterface

> ***Requirement :*** *Compilation and installation of the DataProxy library.*

The **MatlabInterface** folder contains a global **mexme.m** file that compiles all the interfaces.

Compilation steps :

- Open a terminal in **RobotHandler/MatlabInterface/** folder
- Then type in the terminal `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path_to_folder_of_libdataproxies.so` `matlab` `... and run the mexme.m file ...`

Interfaces **mexa** files are in the **RobotHandler/MatlabInterface/** folder. Each time you will launch **matlab** to use this interface you will need to set **LD_LIBRARY_PATH** environment variable :
`export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path_to_folder_of_libdataproxies.so`

NaoAVRecorder

> ***Requirement :*** *OpenCV & QT & sndfile library installed on the computer*

The **NaoAVRecorder** folder contains **CMakeLists.txt** that compiles this C++ application.

Compilation steps :

- Open a terminal in **RobotHandler/NaoAVRecorder/** folder
- Then type in the terminal : `mkdir Build` `cd Build` `cmake .. #` if **DataProxy** library has been installed in the default location `/usr/local/` `cmake .. -DDATAPROXIES_SDK_DIRECTORY=/PATH_TO_DATAPROXY_SDK/ #` if **DataProxy** library has been installed in other location `make`

The executable **naoAVRecorderRelWithDebInfo** is in the **RobotHandler/NaoAVRecorder/Build/** folder.

User Manual

> ***Requirement :*** *Compilation done. cf Compilation Manual*

Steps to follow :

- **Turn Nao on.**
  - Nao's Led should flicker.
  - Nao is ready once he has said : “OGNAK GNOUK”.
  - Nao is alive, it means that motor control is on.
- **Press two times Nao's central button.**
  - Nao goes in a rest mode and set the motor control off.
- **Launch the needed driver(s).**
  - Drivers default path is `RobotHandler/NaoRemoteDriver/Build/bin/`
  - Usage : `./nao<DriverName>DriverRelWithDebInfo NaoIp NaoPort`
  - Can be launch as follow : `./nao<DriverName>DriverRelWithDebInfo`
    - in this case `NaoIp` = "169.254.80.178" and `NaoPort` = 9559.
  - Drivers are ready once you get this message :
    - `STREAMING STARTED` for data streaming drivers
    - `NAO IS WAKE UP AND READY` for motor command drivers
- **Your application can now dialog with the Nao's modules by using the DataProxy library.**

The HeadPoseDemo gives you an example of how to use the DataProxy library in a C++ code. The MatlabSoundLocalizationDemo gives you an example of how to use the DataProxy library in a Matlab code.

NaoAVRecorder

> *Requirement : * Compilation done. cf Compilation Manual*

This tool allows you to record Nao's audio and video on a remote computer.

Launching Step :

- **Turn Nao on.**
- Nao's Led should flicker.
- Nao is ready once he has said : "OGNAK GNOUK".
- Nao is alive, it means that motor control is on.
- **Press two times Nao's central button.**
- Nao goes in a rest mode and set the motor control off.
- **Launch the video and the audio driver.**
- Drivers default path is `RobotHandler/NaoRemoteDriver/Build/bin/`
- Usage : `./naoVideoDriverRelWithDebInfo NaoIp NaoPort`
- Can be launch as follow : `./naoVideoDriverRelWithDebInfo`
    - in this case `NaoIp` = "169.254.80.178" and `NaoPort` = 9559
- Driver is ready once you get this message : `STREAMING STARTED`.
- **Launch the NaoAVRecorder executable.**
- Drivers default path is `RobotHandler/NaoAVRecorder/Build/`
- Usage : `./naoAVRecorderRelWithDebInfo`

Head Pose Demo

> ***Requirement :*** *Compilation done cf [Compilation Manual]*

This example streams the Nao's camera and compute for each frame a face detection algorithm and a head pose estimation algorithm.

Launching Step :

- **Turn Nao on.**
  - Nao's Led should flicker.
  - Nao is ready once he has said : “OGNAK GNOUK”.
  - Nao is alive, it means that motor control is on.
- **Press two times Nao's central button.**
  - Nao goes in a rest mode and set the motor control off.
- **Launch the video driver.**
  - Drivers default path is `RobotHandler/NaoRemoteDriver/Build/bin/`
  - Usage : `./naoVideoDriverRelWithDebInfo NaoIp NaoPort`
  - Can be launched as follow : `./naoVideoDriverRelWithDebInfo`
    - in this case `NaoIp` = "169.254.80.178" and `NaoPort` = 9559
  - Driver is ready once you get this message : `STREAMING STARTED`
- **Launch the HeadPoseDemo executable.**
  - Drivers default path is `RobotHandler/HeadPoseDemo/Build/`
  - Usage : `./naoHeadPoseDemoRelWithDebInfo`

See `main.cpp` - line 37 `DataProxy::getImage(ts,tms,width,height,nbChannels,imgCV.data, DataProxy::BGR);`

Matlab Sound Localization Demo

> ***Requirement*** *: Compilation done. cf Compilation Manual*

This example shows how to launch a simple algorithm to localize a speaker using sound localization and face detection.

Launching Step :

- **Turn Nao on.**
- Nao's Led should flicker.
- Nao is ready once he has said : "OGNAK GNOUK".
- Nao is alive, it means that motor control is on.
- **Press two times Nao's central button.**
- Nao goes in a rest mode and set the motor control off.
- **Launch the video/audio/motion/facedetector driver.**
- Drivers default path is `RobotHandler/NaoRemoteDriver/Build/bin/`
- Usage : `./naoVideoDriverRelWithDebInfo NaoIp NaoPort`
- Can be launch as follow : `./naoVideoDriverRelWithDebInfo` :
  - in this case `NaoIp` = "169.254.80.178" and `NaoPort` = 9559
- Driver is ready once you get this message : `STREAMING STARTED`.
- Set the environment variable :
  - `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path_to_folder_of_libdataproxies.so`
- Launch matlab
- Place the current folder in `RobotHandler/MatlabSoundLocalizationDemo/`
- Execute `speakerLocalization2MIC.m`
- Stop it with the `ctrl+C` command in the matlab terminal.

Developer Manual

NaoRemoteDriver

- Path : RobotHandler/NaoRemoteDriver
- Files :
  - CMakeLists.txt
  - naoutils.h
- Directories :
  - NaoRemoteAudioDriver
  - NaoRemoteFaceDetectorDriver
  - NaoRemoteHeadMotionDriver
  - NaoRemoteMotionDriver
  - NaoRemoteVideoDriver

NaoRemoteAudioDriver

- Path : RobotHandler/NaoRemoteDriver/NaoRemoteAudioDriver
- Files :
  - CMakeList.txt
  - main.cpp
  - naoaudiodriver.cpp
  - naoaudiodriver.h

Comments

This application works with two threads :

- The first one starts the dialog loop and waits for the `q` letter or `ctl+C` signal to cleanly stop the application.
- The second thread is launched by the `NaoAudioDriver` object. To receive sound buffer, we have to overwrite the *process* method from the `AL::ALSoundExtractor` class. Our `NaoAudioDevice` class do so.
  We also have to launch the onboard `AL::ALSoundExtractor` module, it seems that this module is not launched by default.
  The *process* function will pop every *85ms* with the corresponding sound buffers.
  In order to have a non blocking way to receive sound buffer, the `NaoAudioDriver` creates and launchs in a separate thread the
  Some trouble with class fields visibility has been observed, that's why some fields have been moved to global variables.

NaoRemoteFaceDetectorDriver

- Path : RobotHandler/NaoRemoteDriver/NaoRemoteFaceDetectorDriver
- Files :
  - CMakeList.txt
  - main.cpp
  - naofacedetectordriver.cpp
  - naofacedetectordriver.h

Comments

This application works with one thread :

- It starts the dialog loop and waits for `ctl+C` signal to cleanly stop the application.
  The onboard Nao's face detector is running by default.

NaoRemoteHeadMotionDriver

- Path : RobotHandler/NaoRemoteDriver/NaoRemoteHeadMotionDriver
- Files :
  - CMakeList.txt
  - main.cpp
  - naoheadmotiondriver.cpp
  - naoheadmotiondriver.h

Comments

This application works with two threads :

- The first one starts the dialog loop and waits for the `q` letter or `ctl+C` signal to cleanly stop the application.
- The second thread is launched by the `NaoHeadMotionDriver` object (cf method `wakeUp()`). The second thread waits, through a shared mutex, for a motion command. It seems that the `fALMotionProxy->angleInterpolation(fJoinsNames, fAngleLists, fTimeLists, true)` (seee `naoheadmotiondriver.cpp` - l138) is a blocking function.

NaoRemoteMotionDriver

- Path : RobotHandler/NaoRemoteDriver/NaoRemoteMotionDriver
- Files :
  - CMakeList.txt
  - main.cpp
  - naomotiondriver.cpp
  - naomotiondriver.h

Comments

This application works with two threads.

- The first one starts the dialog loop and waits for the `q` letter or `ctl+C` signal to cleanly stop the application.

- The second thread is launched by the `NaoMotionDriver object` (cf method `wakeUp()`).
  The second thread waits, through a shared mutex, for a motion command. It seems that the `fALMotionProxy->angleInterpolationBezier(fJoinsNames, fAngleLists, fTimeLists)` (see `naomotiondriver.cpp - l262`) is a blocking function.

NaoRemoteVideoDriver

- Path : RobotHandler/NaoRemoteDriver/NaoRemoteVideoDriver
- Files :
  - CMakeList.txt
  - main.cpp
  - naovideodriver.cpp
  - naovideodriver.h

Comments

This application works with two threads :

- The first one starts the dialog loop and waits for the `q` letter or `ctl+C` signal to cleanly stop the application.
- The second thread is launched by the `NaoVideoDriver` object (cf method `startCapture()`).
  The second thread pushes in the shared memory new frames (both top and bottom cameras) as soon as possible.

ToDO List

- Check NaoRemoteDrivers -> some uses 2 threads : may be useless.
- Define which data should be send through the shared memory and which should not.
- Add protoBuff -> in the commonfiles part (needed by DataProxy AND NaoRemoteDriver).
- New drivers ??
- AudioVideo synchronized data, according to Isreal request : New driver or proxy ?
- Video : VGA-RGB 15 fps fixed.
- Audio : 4 microphones 48KHz.
- Synchronized format :
- Image delay : 66ms. This define our paquet window.
- getSynchronizedWindow() : return the last image and the 66ms sound buffer that precede this image.