Kinova.API.Jaco

R4.0.5

User guide

Version 1.0.0

January 1 2012

Table des matières

1. Installation guide 3
   1.1. Drivers 3
   1.2. Jacosoft 3
      2.2.1. Install 3
      2.2.2. Uninstall 6
   1.3. API Windows 6
      2.3.1. First installation 6
      2.3.2. Update installation 6
      2.3.3. Uninstall 6
   1.4. API Ubuntu 6
      2.4.1. LIBUSB installation 6
      2.4.2. LIBUSBDOTNET installation 6
      2.4.3. API installation for beginner 7
      2.4.4. API installation for advance user 7
2. Updating your Jaco 7
   2.1. Window 7
   2.2. Ubuntu 9
3. How to use it 9
   3.1. Setting a new project with visual studio 2010 9
   3.2. Examples 10
      4.2.1. Windows 10
      4.2.2. Ubuntu 10
   3.3. Documentation 10
4. Troubleshooting 11
   4.1. Compatible version 11
5. F.A.Q 11

1. Installation guide

1.1. Drivers

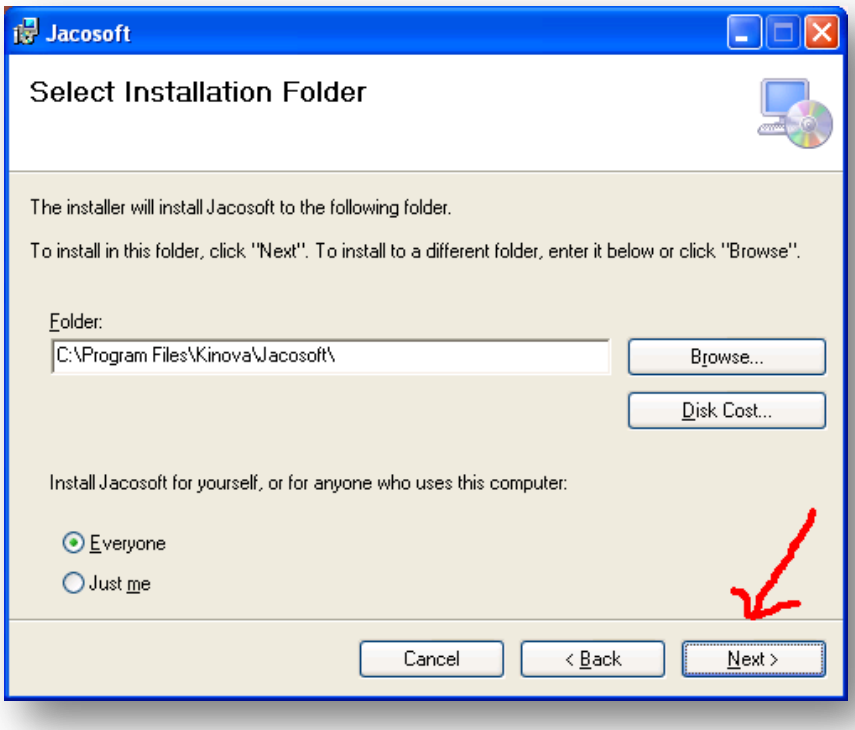
Jaco drivers are located at *RootPackagePath\1 – Jaco Drivers* and are only needed when used on a Windows terminal.

1.2. Jacosoft

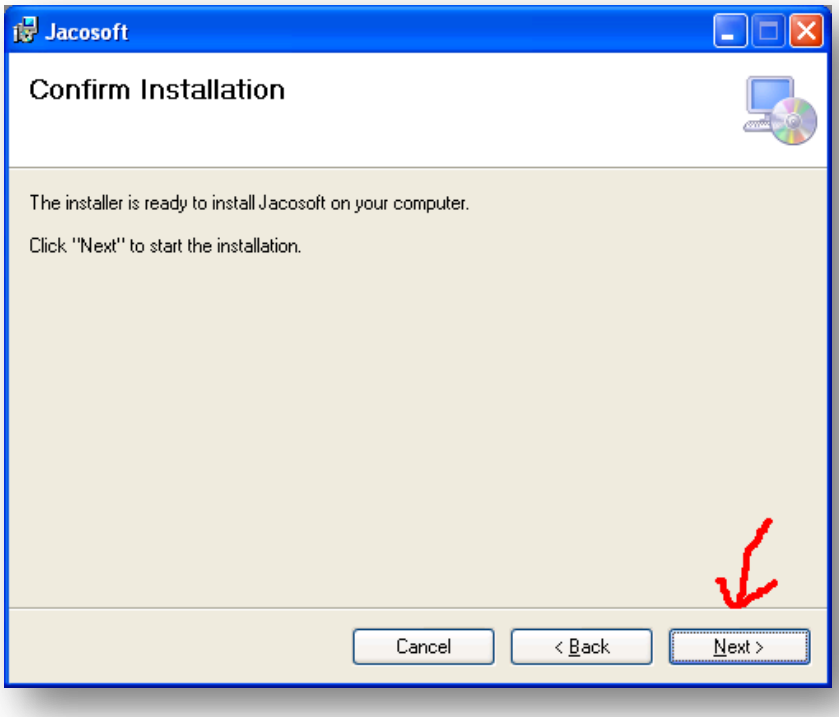
2.2.1. Install

- Install Jacosoft by executing the file *RootPackagePath\2 – Jacosoft[myversion]*\setup.exe and follow the instructions below.
- Click on the NEXT button.

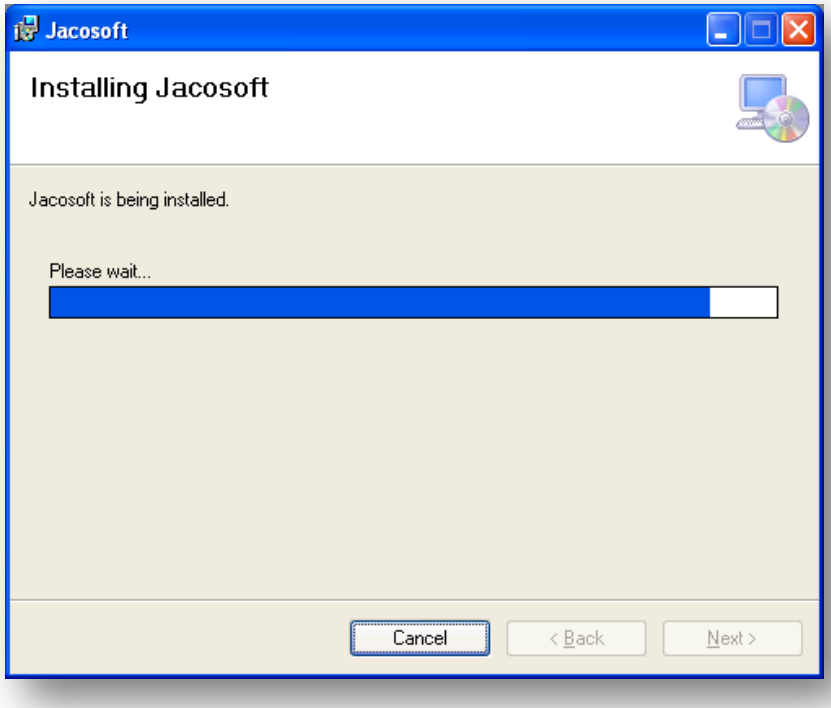
- Choose a location and click on the NEXT button.



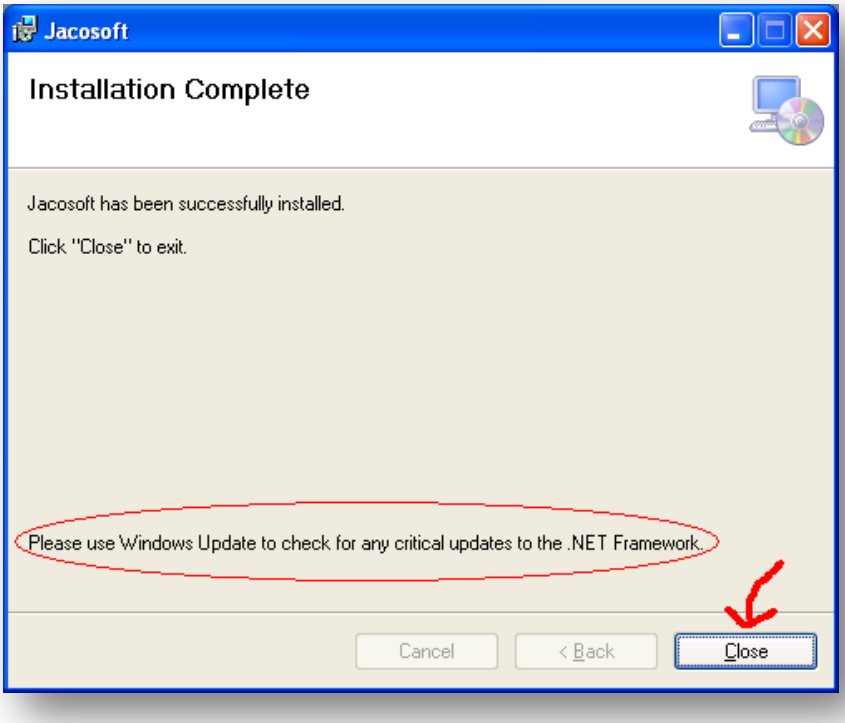
- Confirm the installation by clicking the NEXT button.



- Wait until the end of the installation process.



- Close the installer by clicking the CLOSE button.



2.2.2. Uninstall

- Uninstall Jacosoft via the Window's configuration panel.
- Delete all the files from the license directory located at: *C:\Documents and Settings\YourUserName\Application Data\Kinova\Products\Licenses*.

1.3. API Windows

This section shows how to install the windows version of the API. You have to note that the .net 4 framework is required. The .net 4 framework is free and can be downloaded from Microsoft's web site.

2.3.1. First installation

- Install Jacosoft (see section 2.2.1).
- Get your password that you need to use the API. It is located in the file: *RootPackagePath\4 – API\Password.txt* .
- The installation is completed and you can start to develop an application with the API.

2.3.2. Update installation

- Uninstall Jacosoft via the Window's configuration panel.
- Delete all the files from the license directory located at: *C:\Documents and Settings\Hugo\Application Data\Kinova\Products\Licenses*.
- Reinstall Jacosoft like it was the first time (see section 2.2.1).

2.3.3. Uninstall

- Uninstall Jacosoft via the Window's configuration panel.
- Delete all the files from the license directory located at: *C:\Documents and Settings\Hugo\Application Data\Kinova\Products\Licenses*.

1.4. API Ubuntu

In order to communicate with Jaco you need to install libusb.

2.4.1. LIBUSB installation

In a terminal window, type the command: `sudo apt-get install libusb-1.0-0-dev`

2.4.2. LIBUSBDOTNET installation

- Get the latest binary version of LibUsbDotNet from the website [http://sourceforge.net/projects/libusbdotnet/files/].

- Unzip it!
- Copy the LibUsbDotNet dll to your project directory.

2.4.3. API installation for beginner

If you are an advanced Ubuntu user go directly to the section 2.4.4

If you are not an advanced Ubuntu user you can run the script EasyInstall.sh. The script will create the file *etc/udev/rulesd.d/10-kinova.rules* and write a rule that give the right permissions to libusb. It will also execute the program */InstallationPackage/KinovaUbuntuInstall.exe*

- Run the script EasyInstall.sh

2.4.4. API installation for advance user

I you are an advanced Ubuntu user you can customize your own udev rule to let libusb use USB devices and then execute the script AdvanceInstall.sh. The AdvanceInstall script is exactly the same as the EasyInstall but it skip the udev part. Your custom rule must allow read/write to a device that got a SUBSYSTEM value of : "usb", a DEVTYPE value of : "usb_device" and a idVendor value of : "22cd".

The rule will probably look like :
SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_device",ATTRS{idVendor}=="22cd", MODE="0666"

- Create you customized udev rule.
- Run the script AdvanceInstall.sh

2. Updating your Jaco

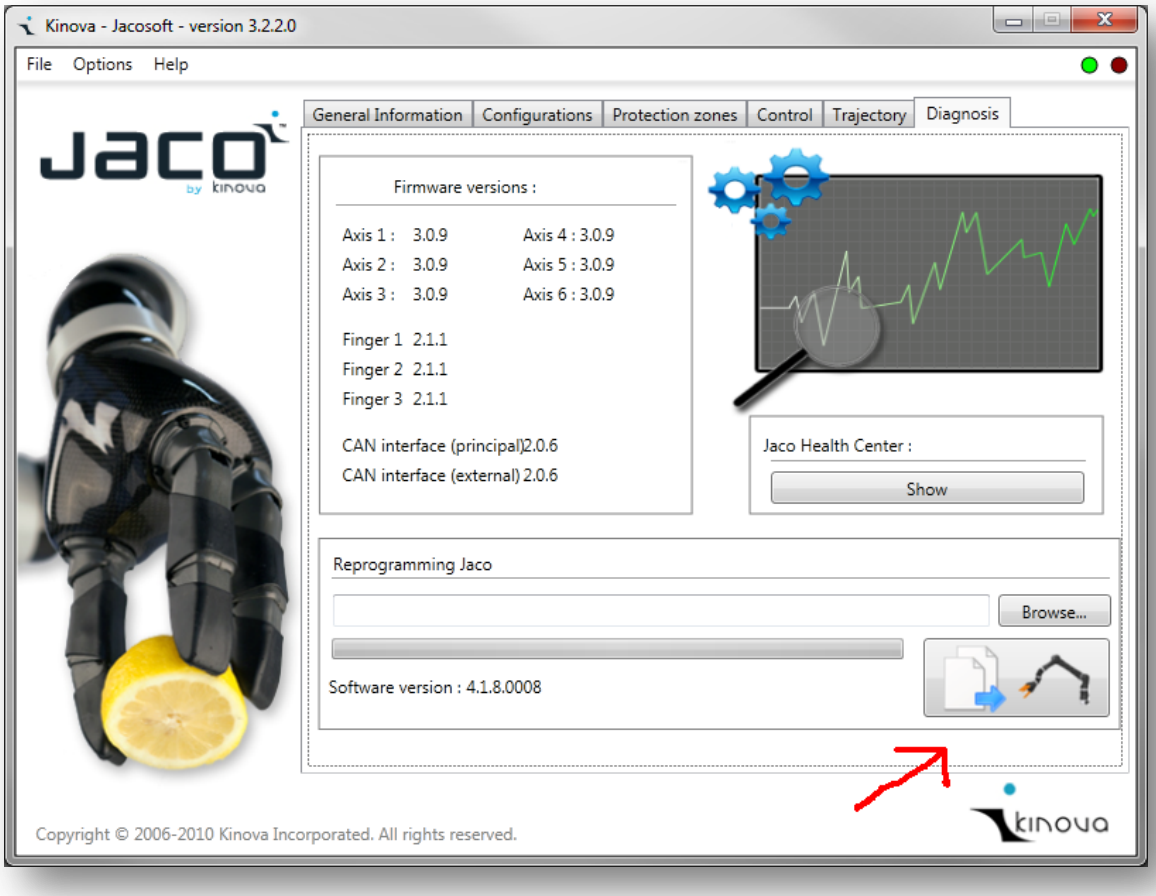
This section shows how to modify the DSP's code of the robotic arm Jaco.

2.1. Window

You need to have Jacosoft installed to continue in this section.

- Open Jacosoft.
- Go to the Diagnosis tab

- Click the BROWSE button.
- Choose the .hex file that represents the new version. Generally, it is located at: RootPackagePath\3 – Jaco dsp code [your version]\.
- Execute the update by pressing the UPDATE button.



2.2. Ubuntu

- Execute the example Kinova.GUI.JacoUbuntu
- Go under the diagnosis tab
- Select the .hex file
- Click the confirmation button
- Boot your Jaco

3. How to use it

3.1. Setting a new project with visual studio 2010

- Open Visual Studio 2010.
- Create a new Visual C# project.

- Build it.
- Copy the directory RootPackagePath\4 – API\Windows [myversion]\DLL\External_DLL\ beside your executable file. Probably in …\Bin\ .
- Add those references to the project. All references are located in RootPackagePath\4 – API\Windows [myversion]\DLL\ .
  - Kinova.API.Jaco.dll
  - Kinova.DLL.ReportBuilder.dll
  - Kinova.DLL.Tools.dllKinova.DLL.Data.dll
  - Kinova.DLL.SafeGate.dll
  - Kinova.DLL.USBManager.dll
- You are ready to code. Don't forget that you need a password to declare a CjacoArm object. That password is located at: *RootPackagePath\4 – API\Password.txt*.

3.2. Examples

4.2.1. Windows

- Open the solution file with visual studio 2010.
- Copy the directory RootPackagePath\4 – API\Windows [myversion]\DLL\External_DLL\ beside your executable file. Probably in …\Bin\ .
- Add those references to the project. All references are located in RootPackagePath\4 – API\Windows [myversion]\DLL\ .
  - Kinova.API.Jaco.dll
  - Kinova.DLL.ReportBuilder.dll
  - Kinova.DLL.Tools.dll
  - Kinova.DLL.Data.dll
  - Kinova.DLL.SafeGate.dll
  - Kinova.DLL.USBManager.dll

4.2.2. Ubuntu

- Open an example project located in either [.../Examples/Kinova.GUI.HealthCenter/] or [.../Examples/Kinova/Kinova.GUI.JacoUbuntu/] with monodevelop.
- Make sure that all DLL references are correct and if they are not, correct them. You need to have a reference to:
- In the method DataInitilization of the class MainWindow, put the good password provided by kinova instead of "MyGoodPassword" at CJacoArm object declaration.
- Execute the project.

3.3. Documentation

API documentation is located at RootPackagePath\4 – API\Windows [myversion]\Documentation and can be accessed via the file Index.html.

Note that the documentation has been generated for Internet Explorer and has not been tested with other browser.

4. Troubleshooting

4.1. Compatible version

API	Jaco's DSP
4.0.3	4.1.8.8
4.0.4	4.1.9.4
4.0.5	4.2.2.12

- To install drivers manually under Windows Vista and Windows 7, type on the command line the command: hdwwiz
- If you can't connect to Jaco, make sure that it is powered on, that you provide a good password at CJacoArm object declaration, that the USB cable is correctly plugged and that Jacosoft is installed.

If you need support of any kind, please contact support@kinova.ca.

5. F.A.Q

What is Jacosoft?

This is an application developed by Kinova to configure and interact with the robotic arm Jaco.

What is the Jaco API?

This is a set of .NET DLL that let you interact programmatically with the robotic arm Jaco.

Is Windows the only O.S. supported by the API?

No.

Ubuntu is also supported.

Do I need other DLL than Kinova.API.Jaco.dll to code and run an application that communicates with the robot arm Jaco?

Yes, but all of them are installed when you install JacoSoft. Either you copy them from your installation package at:

[Package root]/4 – API[…]/Windows/DLL/ for Windows installation

[Package root]/[Ubuntu TAR File]/Ubuntu/Jaco API/ for Ubuntu installation

(See section 4.2 for examples)

Which version of .NET Kinova.API.Jaco is?

Currently, Kinova.API.Jaco is developed under the .NET framework 4.

Can I control (move) Jaco via the API?

Yes.

Basically, there are 2 ways for you to do that task. First, you can use a data structure that emulates a command from the Kinova 3-axis joystick or you can use a data structure that represents a trajectory. There are a couple differences between the 2 options. The joystick emulation is pretty easy to use and very straightforward but your possibilities are limited by those defined in the class CJacoStructures.ControlFunctionnalityValues. The trajectory feature can be trickier to understand at first but it let you use all control features of Jaco.

Joystick emulation

First of all, you need to understand the functionality mapping system completely to use the joystick emulation. Here is a quick reference table to the available functionalities.

ControlFunctionnalityValues	Description
NoFunctionnality	This is a default value that you can use to initialize a variable.
Disable_EnableJoystick	This toggles the ENABLE_DISABLE flag. Basically, it turns the ON/OFF the joystick's control.
Retract_ReadyToUse	Hold down this functionality to let Jaco go to READY position if you are in a NORMAL position, go to RETRACT position if you are in READY position and to get back to READY position if you are in RETRACT position.
Change_TwoAxis_ThreeAxis	That switches the joystick between a 2 axis control and a 3 axis control.
Change_DrinkingMode	This Toggles the drinking mode.
ChangeMode_Left	This functionality let you iterate in the ModeControlsA of your mapping structure.
ChangeMode_Right	This functionality let you iterate in the ModeControlsB of your mapping structure.
DecreaseSpeed	This divides the cartesian speed by 2.
IncreaseSpeed	This multiplies the cartesian speed by 2.
Goto_Position1	This move the robot to the Recorded 1 position if one

	has been stored
Goto_Position2	This move the robot to the Recorded 2 position if one has been stored
Goto_Position3	This move the robot to the Recorded 3 position if one has been stored
Goto_Position4	This move the robot to the Recorded 4 position if one has been stored
Goto_Position5	This move the robot to the Recorded 5 position if one has been stored
RecordPosition1	This let you record your actual position and store it in recorded position 1.
RecordPosition2	This let you record your actual position and store it in recorded position 2.
RecordPosition3	This let you record your actual position and store it in recorded position 3.
RecordPosition4	This let you record your actual position and store it in recorded position 4.
RecordPosition5	This let you record your actual position and store it in recorded position 5.
X_Plus	Once this functionality is held, the robot will move along the X + Axis.
X_Minus	Once this functionality is held, the robot will move along the X - Axis.
Y_Plus	Once this functionality is held, the robot will move along the Y + Axis.
Y_Minus	Once this functionality is held, the robot will move along the Y - Axis.
Z_Plus	Once this functionality is held, the robot will move along the Z + Axis.
Z_Minus	Once this functionality is held, the robot will move along the Z - Axis.
Xteta_Plus	Once this functionality is held, the robot will rotate along the X Axis clockwise.
Xteta_Minus	Once this functionality is held, the robot will rotate along the X Axis counter clockwise.
Yteta_Plus	Once this functionality is held, the robot will rotate along the Y Axis clockwise.
Yteta_Minus	Once this functionality is held, the robot will rotate along the Y Axis counter clockwise.
Zteta_Plus	Once this functionality is held, the robot will rotate along the Z Axis clockwise.
Zteta_Minus	Once this functionality is held, the robot will rotate along the Z Axis counter clockwise.

OpenHandOneFingers	This will open the thumb.
CloseHandOneFingers	This will close the thumb
OpenHandTwoFingers	This will open the hand in 2 fingers mode.
CloseHandTwoFingers	This will close the hand in 2 fingers mode.
OpenHandThreeFingers	This will open the hand in 3 fingers mode.
CloseHandThreeFingers	This will close the hand in 3 fingers mode.

As an example, let says that you are using the 3-axis joystick mapping of Jaco. That means that the functionality X_Plus is mapped with the InclineLR.Plus of the first control mode of the list B. To verify that, you can read the value of your mapping like this :

```
CControlMappingCharts MyCharts = m_Jaco.ConfigurationsManager.GetControlMappingCharts();

int MappingIndex = (int)CJacoStructures.ControlOperator.ExternalJoystick3Axis;

CControlMapping MyMapping = MyCharts.ControlMapping[MappingIndex];

int InclineLRIndex = (int)CJacoStructures.ControlStickValues.InclineLR;

if (MyMapping.ModeControlsB[0].ControlSticks[InclineLRIndex].Plus ==
    CJacoStructures.ControlFunctionnalityValues.X_Plus)
{
       //Here, we know that if I want to go along X + axis I need to use the Plus of the
       InclineLR Stick event of the first mode of the B List.
}
else
{
       //Here, I don't know what is mapped.
}
```

Now that we know what Joystick value to use, we need to create it and send it to Jaco.

```
CJoystickValue MyJoystickValue = new CJoystickValue();

// m_Jaco is a CJacorArm object
m_Jaco.ControlManager.StartControlAPI();

// 1f if you want Plus and -1f if you want Minus.
MyJoystickValue.InclineLR = 1f;

// We send to joystick value to Jaco.
m_Jaco.ControlManager.SendJoystickFunctionnality(MyJoystickValue);

// We wait 2 sec like we were pushing the stick for 2 sec.
Thread.Sleep(2000);

// We modify the joystick value to stop pushing the stick.
MyJoystickValue.InclineLR = 0f;

// We send the modified value like we were releasing the stick.
m_Jaco.ControlManager.SendJoystickFunctionnality(MyJoystickValue);
```

Trajectory control

To use the trajectory features, it can be good to have a small software tool that generate trajectory. The next example is a code snippet that creates and sends a Cartesian trajectory.

```
CPointsTrajectory Trajectory = new CPointsTrajectory();
CTrajectoryInfo Point1 = new CTrajectoryInfo();
CTrajectoryInfo Point2 = new CTrajectoryInfo();
CTrajectoryInfo Point3 = new CTrajectoryInfo();

CPosition tempPosition = new CPosition();

//We move the robot to a cartesian position in space with the joystick
//or any other software solution...

//We get the actual position to create a first trajectory point.
tempPosition = m_Jaco.DiagnosticManager.DataManager.GetPositionLogLiveFromJaco();
Point1.UserPosition = tempPosition.UserPosition;

//Again, we move the robot to a cartesian position in space with the joystick
//or any other software solution...

//We get the actual position to create a second trajectory point.
tempPosition = m_Jaco.DiagnosticManager.DataManager.GetPositionLogLiveFromJaco();
Point2.UserPosition = tempPosition.UserPosition;

//Again, we move the robot to a cartesian position in space with the joystick
//or any other software solution...

//We get the actual position to create a third trajectory point.
tempPosition = m_Jaco.DiagnosticManager.DataManager.GetPositionLogLiveFromJaco();
Point3.UserPosition = tempPosition.UserPosition;

Trajectory.Add(Point1);
Trajectory.Add(Point2);
Trajectory.Add(Point3);

m_Jaco.ControlManager.StartControlAPI();
m_Jaco.ControlManager.SendTrajectoryFunctionnality(Trajectory);
```

Can I control a single actuator (motor, joint, axis) ?

Yes.

To do that, you have 2 options. You can use the 3-axis joystick provided by Kinova or you can use the API in a custom application.

Using the 3-axis joystick

- You set Jaco to angular mode via JacoSoft.
- You move the joystick according to the actuator you want to move.

Using the API and the Joystick command

- You set Jaco to angular mode via the method CJacoArm.ConfigurationManager.SetAngularControl().
- You tell Jaco that from now on the API will be in control via the method CJacoArm.ControlManager.StartControlAPI().
- You send a joystick command via the method CJacoArm.ControlManager.SendJoystickFunctionnality() with the right parameter according to the mapping you are using.

What is the CJacoStructures.ControlMode enum type?

Jaco can be controlled under several modes, this structure define all of them.

How do I know which CControMapping I need to use?

- If you are controlling Jaco via a standard 3-axis joystick made by Kinova, you need to use the ControlMapping[CJacoStructures.ControlOperator.ExternalJoystick3Axis] (index = 0).
- If you are controlling Jaco via a custom joystick that you want to manually map, you need to use the ControlMapping[CJacoStructures.ControlOperator.GUI] (index = 2).
- If you are controlling Jaco via a custom software that you have developed, you need to use the ControlMapping[CJacoStructures.ControlOperator.GUI] (index = 2).
- If you are controlling Jaco via the UniversalAdapter, you need to use the ControlMapping[CJacoStructures.ControlOperator.UniversalAdapter] (index = 4).
- If you want to create your custom mapping, you need to use the ControlMapping[CJacoStructures.ControlOperator.UniversalAdapter] (index = 5).

How do I know if the communication between my application and Jaco is broken?

Use the method CJacoArm.JacoIsReady(). It returns a boolean value that indicates if yes(true) Jaco is online or no(false) it is not.

I've modified the protection zone list of Jaco and nothing happened, why?

Every time you modify the protection zone list of Jaco, you need to reboot the arm. Protection zone is a complex feature and you should make sure you understand it perfectly before you use it. If for some reason you need to erase all protection zone inside Jaco, you can call the method CJacoArm.ConfigurationsManager.DeleteAllProtectionZones() and then reboot the robot.

Can I set the MaxLinearSpeed / MaxAngularSpeed anytime while my application is running?

Of course, but keep in mind that your value must be within a specific range defined by the const value in the class Jaco.CThreshold from the namespace Kinova.DLL.Data. For example, the MaxLinearSpeed must be between CThreshold.LINEAR_SPEED_MAX_LOWER(0.0f) and CThreshold.LINEAR_SPEED_MAX_UPPER(0.15f). If the value is not in this range, Jaco's client configuration will not be modified.

What the IncreaseSpeed / DecreaseSpeed functionalities do exactly?

IncreaseSpeed will double the speed while DecreaseSpeed will divide it by 2. Of course, you cannot exceed the max value configured in Jaco and you cannot go below 0.0f.

I can get the position of the hand but I don't know what the reference is.

In Cartesian mode, the position is represented by a CVectorEuler object. It contains 2 float arrays named Position and Rotation. Both arrays have 3 elements (You can use the NB_POSITION and NB_ROTATION constant). The Position array contains the translation of your position and the Rotation array contains the orientation of the hand. Units are meters and Rad.