USER MANUAL

XARM LEWANSOUL ROS INTEGRATION

Date of issue: November 28, 2023

Release Version: F_improving_docs

daira

Dynamic AI Robotic Agents

Overview

1 Preface

This is the documentation for ROS integration of the Lewansoul-Hiwonder xArm.

Here you will find:

- Low level drivers to comunicate with the robot either in python or C++
- Hardware interface to control the robot with either position or trajectory control
- Integration with RVIZ and Moveit! including URDF and SRDF files which allow you to plan trajecto-ries

---

Note: At the moment the gripper has not been yet integrated.

---

2 Overview

- *Quick Start*

2.1. Quick Start

2.1.1. Installation

System requirements

- Ubuntu 18.04
- ROS Melodic

Install package dependencies

```
$ sudo apt-get install libhidapi-hidraw0 libhidapi-libusb0
$ pip install --user hid
$ sudo apt-get install python-dev libusb-1.0-0-dev libudev-dev
$ sudo pip install --upgrade setuptools
$ sudo pip install hidapi
```

Install HIDAPI

1. Clone hidapi from https://github.com/libusb/hidapi
2. Install dependencies as mentioned in hidapi repository:

   ```
   $ sudo apt-get install libudev-dev libusb-1.0-0-dev libfox-1.6-dev
   $ sudo apt-get install autotools-dev autoconf automake libtool
   ```

3. Run all the commands to build as mentioned in the readme of the repository:

   ```
   $ ./bootstrap
   $ ./configure
   $ make
   $ sudo make install
   ```

Build xArm_Lewansoul_ROS

1. Clone this repository

   ```
   $ git clone https://github.com/diestra-ai/xArm_Lewansoul_ROS.git
   ```

2. Build

   ```
   $ catkin_make
   ```

2.1.2. Run

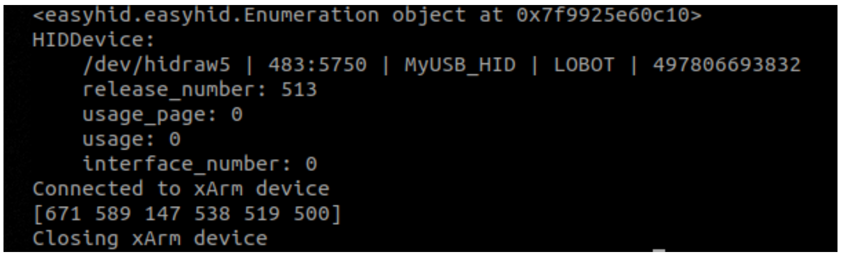
1. Connect the robot to any USB port of your computer
2. Turn the robot on. You should see the pink and blue lights on the control card.
3. To control the robot with code you sould be in sudo mode

   ```
   $ sudo -s
   ```

4. Run the following script to check the computer is able to detect and control the robot. You should be in the folder of xArm_Lewansoul_ROS

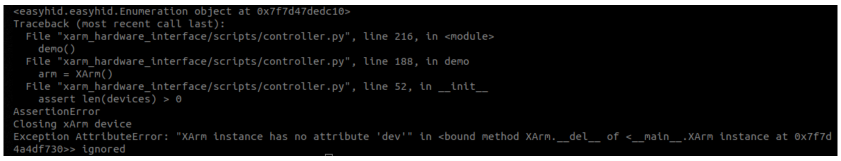
   ```
   $ python xarm_hardware_interface/scripts/controller.py
   ```

   - If the robot is correctly detected you should see the following output:

     ```
     <easyhid.easyhid.Enumeration object at 0x7f9925e60c10>
     HIDDevice:
         /dev/hidraw5 | 483:5750 | MyUSB_HID | LOBOT | 497806693832
         release_number: 513
         usage_page: 0
         usage: 0
         interface_number: 0
     Connected to xArm device
     [671 589 147 538 519 500]
     Closing xArm device
     ```

   - The robot should move to a vertical position and then open and close the gripper followed by rotating its second and third joints.
   - If the robot is not detected the following output would appear:

     ```
     <easyhid.easyhid.Enumeration object at 0x7f7d47dedc10>
     Traceback (most recent call last):
       File "xarm_hardware_interface/scripts/controller.py", line 216, in <module>
         demo()
       File "xarm_hardware_interface/scripts/controller.py", line 188, in demo
         arm = XArm()
       File "xarm_hardware_interface/scripts/controller.py", line 52, in __init__
         assert len(devices) > 0
     AssertionError
     Closing xArm device
     Exception AttributeError: "XArm instance has no attribute 'dev'" in <bound method XArm.__del__ of <__main__.XArm instance at 0x7f7d
     4a4df730>> ignored
     ```

     in this case check that the robot is on and that you are running from sudo mode, or try another USB port or cable. Also maybe restart the system.

     With this python file you can use different functions to control the robot.

5. Control the robot using ROS

   To have the robot running in ROS, launch the following

   ```
   $ roslaunch xarm_launch xarm.launch
   ```

3 Software Description

3.1. Driver Description

3.1.1. USB Protocol

A repository describing the communication protocol was found here: https://github.com/ccourson/LewanSoul-xArm.

From there, we took the information about the protocol to communicate with the robot.

A packet transmitted to the xArm will have the following format:

	id	header	length	command	parameters
Bytes	1	2	1	1	0 or more
Comments	Any number.	Always 0x5555.	Here to end.	See commands.	See commands.

Commands are essentially request packets embedded into USB HID reports. Requests and responses are described in the following syntax:

- Each field is seperated by a space.
- Each field is described by the type in parentheses.
- Curly braces denote that their content may be repeated more than once.

The following are the set of available requests:

```
ServoMove             3  (byte)count (ushort)time { (byte)id (ushort)position }
GroupRunRepeat        5  (byte)group[255=all] (byte)times
GroupRun              6  (byte)group (ushort)count[0=continuous]
GroupStop             7  -none-
GroupErase            8  (byte)group[255=all]
GroupSpeed           11  (byte)group (ushort)percentage
xServoOffsetWrite    12  *** not sure
xServoOffsetRead     13  *** not sure
xServoOffsetAdjust   14  *** not sure
GetBatteryVoltage    15  -none-; (ushort)millivolts
ServoOff             20  (byte)count { (byte)id }
ServoPositionRead    21  (byte)count { (byte)id }; (byte)count { (byte)id (ushort)position }
ServoPositionWrite   22  (byte)count { (byte)id (ushort)position }
```

(continues on next page)

(continued from previous page)

```
ServoOffsetRead      23  (byte)count { (byte)id }; (byte)count { (byte)id (sbyte)offset }
ServoOffsetWrite     24  (byte)id (sbyte)offset
BusServoMoroCtrl     26  (byte)id (byte)??? (ushort)speed
```

Taking this information, we have created files to control the robot in python and c++. We have subsequently used the c++ implementation to create the following robot interfaces but we have left the python one in case it is helpful for someone.

Python driver

We found the following implementation in python for this robot (Author: Maxime Chevalier-Boisvert). We modified it and you can find it here.

To run it, turn the robot on and run the following:

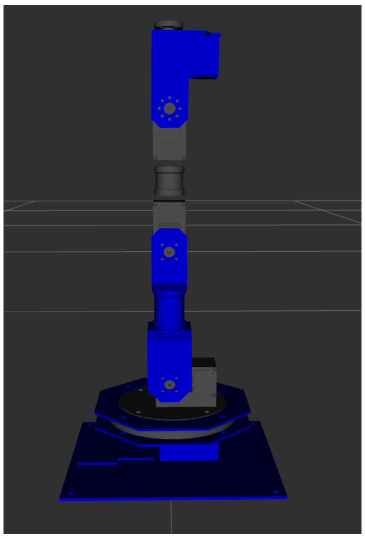
```
$ sudo -s
$ python xarm_hardware_interface/scripts/controller.py
```

There are some commands that will be executed as examples of how to control the robot. You should see the robot doing some movements.

C++ driver

Starting from the python implementation, we have created the c++ version of it that you can find here

Since there is not information about the joint limits in the robot manual, we have manually found out the values in units by using the Lewansoul mobile app. Then, we calculated their equivalent in radians setting the zero when the robot is in the vertical position as showing in the following picture:



Joint	Units			
	Min	Max	-pi/2	pi/2
2	50	995	200	980
3	100	950	140	880
4	50	950	130	870
5	135	950	140	880
6	50	900	90	845

Joint	Radians	
	Min	Max
2	-2.17494876	1.63121157
3	-1.740612146	1.86797401
4	-1.910427965	1.910427965
5	-1.592023304	1.86797401
6	-1.737238322	1.79965407

3.2. Hardware Interface

We used this following tutorial to create the ros_control interfaces for this robot. You can find our implementation here.

3.3. URDF

Meshes were taken from https://grabcad.com/library/lewansoul-6dof-robot-arm-1. Different links were exported as single files using Solidworks. The origin and orientation were changed using Blender according with the axis of rotation of the real robot. Joints were defined in the URDF file.

Warning: At the moment, there is no information about inertia.

3.4. RVIZ-MoveIt!

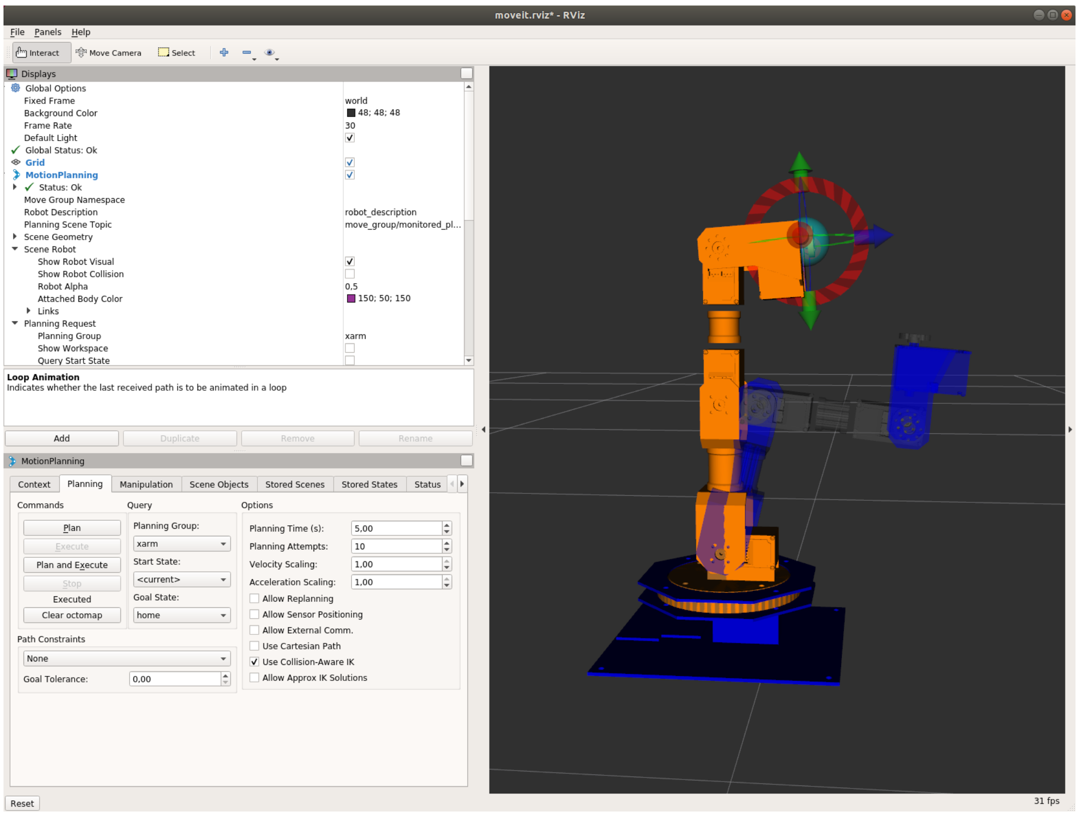
Running `xarm.launch` file will launch MoveIt! and RVIZ integration that will allow you to plan trajectories. In this case the joints are controlled using trayectory controller.

```
$ roslaunch xarm_launch xarm.launch
```

We have integrated xArm with MoveIt! using MoveIt! Setup Assistant. Here. you can find the Moveit! configuration and the srdf file here.

Note: You can control the robot using only position control using the following launch file:

```
$ roslaunch xarm_hardware_interface xarm_position_controller.launch
```



Global Options	
Fixed Frame	world
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: Ok	
Grid	✓
MotionPlanning	✓
Status: Ok	
Move Group Namespace	
Robot Description	robot_description
Planning Scene Topic	move_group/monitored_pl...
Scene Geometry	
Scene Robot	
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	0.5
Attached Body Color	■ 150; 50; 150
Links	
Planning Request	
Planning Group	xarm
Show Workspace	☐
Query Start State	☐

Loop Animation
Indicates whether the last received path is to be animated in a loop

Planning Group: xarm
Start State: <current>
Goal State: home

None
Goal Tolerance: 0,00

Planning Time (s): 5,00
Planning Attempts: 10
Velocity Scaling: 1,00
Acceleration Scaling: 1,00
☐ Allow Replanning
☐ Allow Sensor Positioning
☐ Allow External Comm.
☐ Use Cartesian Path
✓ Use Collision-Aware IK
☐ Allow Approx IK Solutions