[Github Guide](https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/master/ur_robot_driver/doc/usage_example.md)

#### The setup

The setup will take you through building the necessary software on your device, specifically the *ur\_robot\_driver*, that will be then used in controlling the robot arm.

Prerequisites: ROS I must be installed on your device (this tutorial works with ros noetic).

* Instructions for installing ros noetic can be found here <http://wiki.ros.org/noetic/Installation/Ubuntu>

1. Begin by creating a new directory to store the driver.

* *mkdir catkin\_ws/src*

1. Then gointo catkin\_ws/src and clone the repository containing the driver

* *git clone* [*https://github.com/UniversalRobots/Universal\_Robots\_ROS\_Driver*](https://github.com/UniversalRobots/Universal_Robots_ROS_Driver)

1. Return to main catkin\_ws space and being building the workspace

* *catkin build*

1. If an error occurs because you are missing dependencies, run the following

* *rosdep update*
* *sudo apt update*

Your catkin\_ws directory should now be expanded to include the necessary files to use the driver.

The driver in the catkin workspace allows for communication between your device and the teach pendant. The catkin workspace allows for the driver to proceed without any limitations due to absent dependencies.

My notes:

* *catkin* is the build package (or build manager) for ROS I, while *colcon* is build package for ROS I
  + A build manager aggregates all the dependencies to be used in following projects
  + Doing *catkin build* will then set up the workspace and ensure the creation and sourcing of all necessary packages.
* The ros driver in the catkin workspace allows for communication between computer and teach pendant. The catkin workspace simply allows for the driver to proceed without any hiccups or limitations.
* Useful command: *vim <file\_name>* | looks into files

#### Establish Connection with Teach Pendant via Robot Driver

Now that you have the driver properly installed, you may now begin a connection with the ur\_robot arm. This will be done through instructions on the teach pendant as well as running command lines to activate the ur\_robot\_driver.

1. Turn on the Teach Pendant
2. Turn on the robot arm
3. Connect a ethernet port to your device
4. Start the robot driver from terminal, this will begin with going into the *catkin\_ws* directory and sourcing the necessary dependencies

* source /opt/ros/noetic/setup.bash
* source devel/setup.bash
* Note: the first command will source the underlays and the second command will source the overlay, *devel*

1. Once the necessary workspace has been created, you may now launch the driver
   * *roslaunch ur\_robot\_driver <robot\_name>\_bringup.launch robot\_ip:=<ip\_addres>*
   * Note: this lab will have the following configuration
     + <robot\_name> : *ur5e*
     + *<*ip\_address> : *192.168.0.101*
2. Go to the *Program* tab on the pendant, expand the tree *Robot Program*. Notice it is empty cause no connection with a computer has been established yet.
   * First go into the *Installation* tab → *URCaps →* change *Host IP* and *Host Name* to your device’s IP address.
   * Then back to *Program* → *URCaps* →click on *External Control*. Now a leaf will be added to tree that indicates ip address of your device

My Notes:

* In starting up the driver, JMoon created his own launch file that configured his custom specifications.
* The *Host IP* and *Host Name* of the Legion laptop are both 192.168.0.103
* Jmoon’s laptop is 192.168.0.102

#### Control Robot using Programs

Once a connection has been established, you can now control the arm’s movement. This can be done in one of two ways. The first method involves using python scripts and the terminal, whilst the second utilizes MoveIt!, a program that allows for freer motion of play

Note: this tutorial will not have instructions on how to make the scripts, but will work with them.

##### 

##### Using Python Scripts

1. Before running any scripts, first press the play button on the teach pendant -> ‘play from selection’
2. If you have a script already finished and ready to use, then run the following line from a new terminal window.

* *rosrun ur\_robot\_driver <script\_path>*
* Make sure the script is accessible in the directory you are in.

1. Otherwise, the *ur\_robot\_driver* provides a *test\_move* script that can be run with

* *rosrun ur\_robot\_driver test\_move*

##### Using MoveIt

* See the

# To do

Each method has its own benefits. Using a python script is more concrete and should be used for longer running and complex movements. However, it does not allow for a pre-visualized simulation of the arm’s movement as provided by MoveIt!. MoveIt! is best used for visualizing the arm’s movement for simpler motions and if you want to ‘play’ around.

#### 

My Notes:

* MoveIt basically helps find a path to move robot arm according to specified waypoints
* Nice to help visual the robot arm’s movements for a given coordinate. Say for example, the waypoints in the test\_move file

#### (TO DO) Visualize the Robot Using RVis

* Rvis reads from the topics and extends them to your computer (it is a subscriber)
  + For instance, it will communicate the joint positions to your computer
* Roslaunch —-- (find in github)
* Now you have a visual representation

# 

#### General Notes

* Underlays and overlays
* Underlays are located in /opt/ros/noetic or /opt/ros/foxy
* In order to use ROS 1, run command source /opt/ros/noetic/setup.bash
  + To use ROS 2 run source /opt/ros/foxy/setup.bas
  + The setup.bash
* Doing sudo apt install downloads the package into the underlay
* Rosdep install will automatically find the dependencies needed for a build
  + Sudo apt install is used when you want to manually install each package yourself
* Underlays are sourced in the overlays to use in more specific applications.
  + Think underlays as tools and overlays as projects that use those tools.
  + Before you begin a project, you need to first obtain the tools you are going to use. Failure to do so will inhibit progress in the overlay. Obtaining the tools is sourcing the dependencies
* To indicate a directory as a ros package, need to include CMakeLists.txt and package.xml file in the directory.
  + These files work to generate the package
* Roscd allows you to cd specifically into ros packages, while cd is just the general change directory command
  + Before you roscd into a ros package, you must first source that package so that it is available in that overlay.