

# Installation Guide for Running Robotito using SCS

David Ehrenhaft

University of South Florida

## Introduction

---

The following guide will provide a step-by-step series of instructions to install all software components necessary to run the Robotito robot using the SCS framework on Ubuntu.

Note: This guide will require any account with root access. If you are NOT an administrator, contact one to install these components.

## Installing SCS

---

Before continuing, you should install and run SCS v2 (refer to [this guide](#) for instructions).

Note: When cloning SCS, you must clone the branch names "v2-david". Use the following command in place of the ordinary "git clone" command:

```
git clone --single-branch -b v2-david https://github.com/biorobaw/scs.git
```

Although it isn't necessary, it is recommended to set up SCS to work from Eclipse (refer to [this guide](#)).

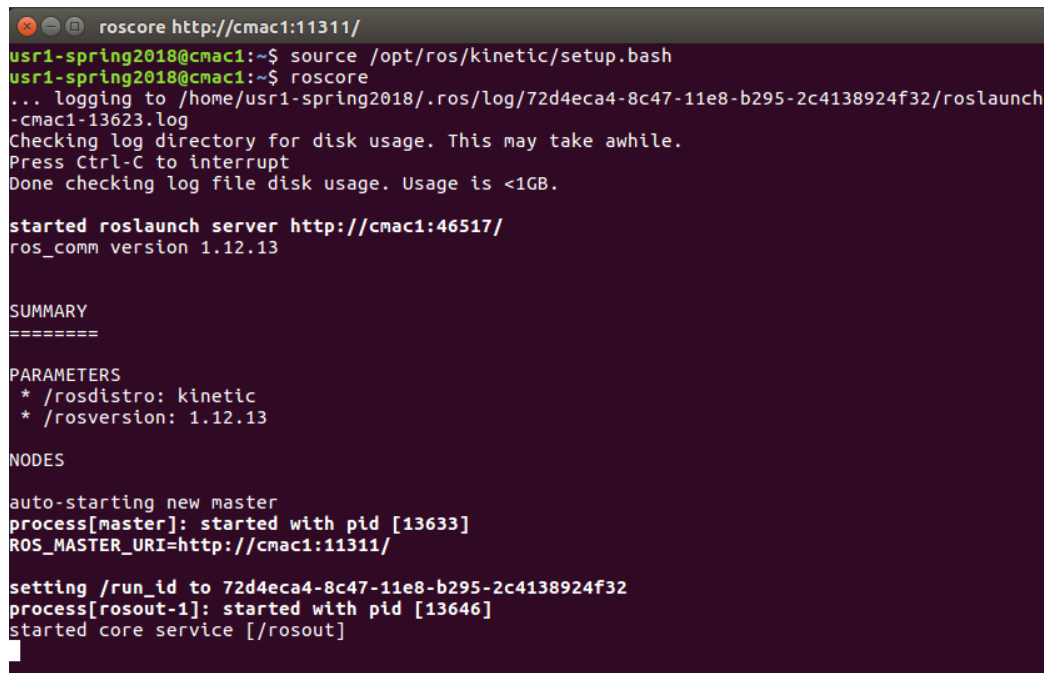
## Setting up the ROS environment

---

The camera system, pose detector for the robot, and pose detector for the walls all run as ROS nodes. The instructions below are necessary to run these components.

Note: In the sections below, any strings contained in quotes are terminal commands.

- 1) Install ROS (I have tested the code using the Kinetic distro)
  - a. Ros should be installed following the installation guide at <http://wiki.ros.org/ROS/Tutorials>. It is suggested that you download the full desktop version.
  - b. Once installed, you should be able to run ROS commands in a terminal after you sourcing ROS, for instance “roscore” (see Figure 1 below).
  - c. You can source ROS by entering “source /opt/ros/kinetic/setup.bash” in each terminal you wish to use ROS in. You can also add this line to your bash file to source ROS automatically whenever you open a terminal.



```
roscore http://cmacl:11311/
usr1-spring2018@cmacl:~$ source /opt/ros/kinetic/setup.bash
usr1-spring2018@cmacl:~$ roscore
... logging to /home/usr1-spring2018/.ros/log/72d4eca4-8c47-11e8-b295-2c4138924f32/roslaunch
-cmacl-13623.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://cmacl:46517/
ros_comm version 1.12.13

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.13

NODES

auto-starting new master
process[master]: started with pid [13633]
ROS_MASTER_URI=http://cmacl:11311/

setting /run_id to 72d4eca4-8c47-11e8-b295-2c4138924f32
process[rosout-1]: started with pid [13646]
started core service [/rosout]
```

Figure 1: Proper execution of "roscore" command; ROS is properly installed.

- 2) Create a catkin workspace to run custom ROS nodes. Note that catkin is included with the full desktop install of ROS.
  - a. In a terminal that has sourced ROS,
    - i. “mkdir -p ~/catkin\_ws/src”
    - ii. “cd ~/catkin\_ws/”
    - iii. “catkin\_make”
  - b. Once complete, the make command should have completed without errors, and there should be src, devel, and build folders in catkin\_ws.
  - c. Refer to this guide for full instructions and explanations if the above doesn’t work: [\[http://wiki.ros.org/catkin/Tutorials/create\\_a\\_workspace\]](http://wiki.ros.org/catkin/Tutorials/create_a_workspace).

- 3) Clone the Robotito files into your catkin workspace:
  - a. In a terminal, “cd catkin\_ws/src”
  - b. “git clone --single-branch -b SCSCCompatible <https://github.com/biorobaw/robotito.git>”
  - c. “cd ..” (return to catkin\_ws/)
  - d. “catkin\_make”
  - e. The make command should complete without errors.
  
- 4) You may need to install the ROS code for the Camera 1394 package
  - a. In a terminal, “cd catkin\_ws/src”
  - b. “git clone <https://github.com/ros-drivers/camera1394.git>”
  - c. You may also need to install the libdc1394-22 library
    - i. “sudo apt-get install libdc1394-22-dev”
  - d. “cd ..” (return to catkin\_ws/)
  - e. “catkin\_make”
  - f. The make command should complete without errors.
  
- 5) You should now be able to run the pose detector ROS code
  - a. In a terminal, “roscore” (remember to source ROS using “source /opt/ros/kinetic/setup.bash”)
  - b. In a new terminal,
    - i. “source /opt/ros/kinetic/setup.bash”
    - ii. “source /catkin\_ws/devel/setup.bash”
    - iii. “roslaunch pose\_detector.launch”
    - iv. The code should run without crashing or errors. To test, try “rostopic list” in another new terminal – multiple topics should appear (see Figure 2 below).

```

/home/usr1-spring2018/catkin_ws/src/robotito/launch/pose_detector.launch http://l...  usr1-spring2018@cmac1: ~
usr1-spring2018@cmac1:~$ source /opt/ros/kinetic/setup.bash
usr1-spring2018@cmac1:~$ source catkin_ws/devel/setup.bash
usr1-spring2018@cmac1:~$ roslaunch robotito pose_detector.launch
... logging to /home/usr1-spring2018/.ros/log/72d4eca4-8c47-11e8-b295-2c413897...
-cmac1-14510.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: Traditional processing is deprecated. Switch to --inorder processing!
To check for compatibility of your document, use option --check-order.
For more infos, see http://wiki.ros.org/xacro#Processing_Order
xacro.py is deprecated; please use xacro instead
started roslaunch server http://cmac1:36076/

SUMMARY
=====

PARAMETERS
* /robot_description: <?xml version="1...
* /roscdistro: kinetic
* /rosversion: 1.12.13
* /stingray1/ar_track_alvar1/tf_prefix_ar: stingray1/
* /stingray1/auto_gain: 3
* /stingray1/auto_gamma: 0
* /stingray1/auto_saturation: 3
* /stingray1/auto_shutter: 3
* /stingray1/camera_info_url: package://robotit...
* /stingray1/frame_id: stingray1
* /stingray1/gain: 150

usr1-spring2018@cmac1:~$ rostopic list
/clicked_point
/diagnostics
/initialpose
/joint_states
/move_base_simple/goal
/robotito/pose
/rosout
/rosout_agg
/stingray1/ar_pose_marker
/stingray1/ar_track_alvar1/enable_detection
/stingray1/ar_track_alvar1/parameter_descriptions
/stingray1/ar_track_alvar1/parameter_updates
/stingray1/camera_info
/stingray1/image_raw
/stingray1/image_raw/compressed
/stingray1/image_raw/compressed/parameter_descriptions
/stingray1/image_raw/compressed/parameter_updates
/stingray1/image_raw/compressedDepth
/stingray1/image_raw/compressedDepth/parameter_descriptions
/stingray1/image_raw/compressedDepth/parameter_updates
/stingray1/image_raw/theora
/stingray1/image_raw/theora/parameter_descriptions
/stingray1/image_raw/theora/parameter_updates
/stingray1/parameter_descriptions
/stingray1/parameter_updates
/stingray1/visualization_marker
/stingray1/visualization_marker_array
/stingray2/ar_pose_marker
/stingray2/ar_track_alvar2/enable_detection
/stingray2/ar_track_alvar2/parameter_descriptions
/stingray2/ar_track_alvar2/parameter_updates
/stingray2/camera_info
/stingray2/image_raw
/stingray2/visualization_marker
/tf
/tf_static
/walls

```

Figure 2: Proper pose detector launched and proper rostopic list results

Note: The code in step (5) requires the Stingray cameras to be connected to the host machine. If you get an error, use Coriander (in a terminal enter “coriander” to open) to ensure that cameras are connected to the system. If not, you must create a new launch file that receives the video stream in some other way.

## Setting up the code for SCS extension

---

To communicate between SCS and the Robotito, additional libraries must be installed.

- 1) Install [libxbee3 library](#)
  - a. In any directory outside of your catkin workspace or SCS,
    - i. “git clone <http://github.com/attie/libxbee3.git>”
  - b. In a terminal, change the directory to libxbee3/ and build the file:
    - i. “make configure”
    - ii. “make all”
    - iii. “sudo make install”
  
- 2) You will probably need to manually install the RXTX Serial transmission library.
  - a. Download the binaries for version 2.2pre2(prerelease) from <http://rxtx.qbang.org/wiki/index.php/Download>.
  - b. Unzip the binaries, then move to that directory in a terminal. You need to move RXTXcomm.jar to your java libraries:
    - i. “sudo cp RXTXcomm.jar \$JAVA\_HOME/jre/lib/ext/”
  - c. Additionally, you’ll need the librxtxSerial.so file. Change the directory to x86\_64... folder in the binaries download:
    - i. “sudo cp librxtxSerial.so \$JAVA\_HOME/jre/lib/amd64/”

Note: You may disregard the “WARNING: RXTX Version mismatch” statement when you run the software.

## Setting up the Robotito’s firmware

---

Separate software is needed to upload the Robotito’s firmware and to handle Xbee configurations.

- 1) Install Arduino Studio
  - a. Go to <https://www.arduino.cc/en/Main/Software> and select the appropriate version to download. You will be redirected to a donation page, but you can skip the donation and continue.
  - b. Unzip the archive, then enter “./install.sh” in a terminal in the extracted directory to install the program.
  - c. For a more comprehensive guide: <https://www.arduino.cc/en/Guide/Linux>.

- 2) Install the Teensyduino add-on
  - a. Download the file from [https://www.pjrc.com/teensy/td\\_download.html](https://www.pjrc.com/teensy/td_download.html)
  - b. Open a terminal to the directory where the file was downloaded. You will need to make it an executable by entering “chmod 755 TeensyduinoInstall.linux32” or “chmod 755 TeensyduinoInstall.linux64”
  - c. Enter “./TeensyduinoInstall.linux32” or “./TeensyduinoInstall.linux64” to open the installer, then select the directory where you installed Arduino Studio.
  
- 3) Upload the firmware to the Robotito
  - a. Go to `catkin_ws/src/robotito/firmware/` and find the file “firmware.ino”. Connect the robot to the current computer using a USB b cable and upload the firmware.
  
- 4) Install XTCU software to change Xbee parameters or to test the modules
  - a. Download the software from <https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu>
  - b. In a terminal in the directory where the file was downloaded, “chmod +x 40002881\_R.run”
  - c. Enter “./40002881\_R.run” to run the installer

Note: The filename for the XTCU download may not be the same as in steps b. and c. above.

- 5) Set the profiles for both Xbee modules
  - a. Plug in the first Xbee using a USB cord and an Xbee platform.
  - b. Open XTCU and add the Xbee by selecting its port.
  - c. In the first tab, select to load a profile, go to `/SCS/Documentation/` and select “SCS\_Xbee\_Coordinator\_Profile.xpro”.
  - d. Repeat steps a-c for the second Xbee but select the “SCS\_XBEE\_Receiver\_Profile.xpro” instead.

Note: You will need to adjust the MAC addresses for the profiles to match the Xbees used.

- 6) Confirm that Xbee communication works as expected
  - a. Place a battery in the robot, make sure the receiver Xbee is placed in the Robotito and the other Xbee is attached to the computer.
  - b. In XTCU, add the Coordinator Xbee and go to the second tab. Lock the connect, and multiple packets should be received from the Receiver Xbee.
  - c. If desired, you can create a control packet following the format “v 80 80 80” to make the Robotito stop, and “v 90 80 80” to make the Robotio move forward (continuously).

You should now be able to run the Robotito using SCS. Refer to the Run Guide to learn how to run the complete setup.