**OpenROV and ROS Integration**

**Introduction**

This document describes how we integrated ROS (Robot Operating System) with OpenROV and how to setup and use the integrated systems.

The electronics of the OpenROV consist of an Arduino connected to a Beaglebone Black. The Beaglebone Black runs the native OpenROV software that uses node.js and communicates with JSON messages.

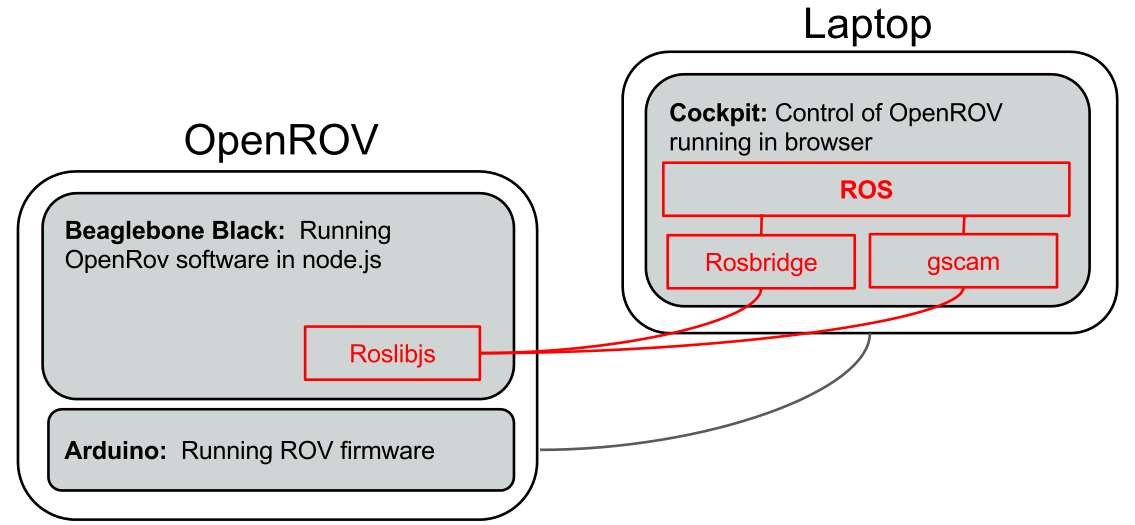
Since the ROV is tethered to a laptop and the Beaglebone Black has limited computing resources, we decided to implement ROS on the laptop instead of directly on the ROV. In this setup, the ROS Master runs on the laptop and interacts with two ROS nodes that we have installed: Rosbridge and gscam. The ROV contains an additional ROS node (Roslibjs) which interacts with Rosbridge for data messages.

Rosbridge provides an API that can translate JSON messages into ROS messages. It acts as a ROS node that can publish or subscribe to messages. We communicated with it via a websocket connection and roslibjs.

Roslibjs interacts with Javascript to publish or subscribe to ROS messages. It uses websockets to communicate with Rosbridge.

Gscam is a ROS package that imports video and converts it into a format that ROS can easily work with. It uses gstreamer which provides it with image processing tools.

The figure below shows the basic structure of the ROS to OpenROV integration.



The ROS to OpenROV system requires installing software on the OpenROV Beaglebone and the Linux system.

The major steps for installing the ROS to OpenROV integration are:

On the Beaglebone:

1. Setup SSH..
2. Install dependencies (apt-get) for roslibjs.
3. Install other dependencies (npm) for roslibjs.
4. Install roslib.
5. Load the ros plugin.

On the Linux computer:

1. Install ROS.
2. Install ROSBridge.
3. Install gscam.
4. Install OpenROV-ros

**ROV Setup**

Setting up the ROV involves installing The dependencies for the OpenROV plugin and then cloning the plugin onto the ROV. The main dependency for the ROV plugin is roslibjs, however roslibjs has several dependencies as well.

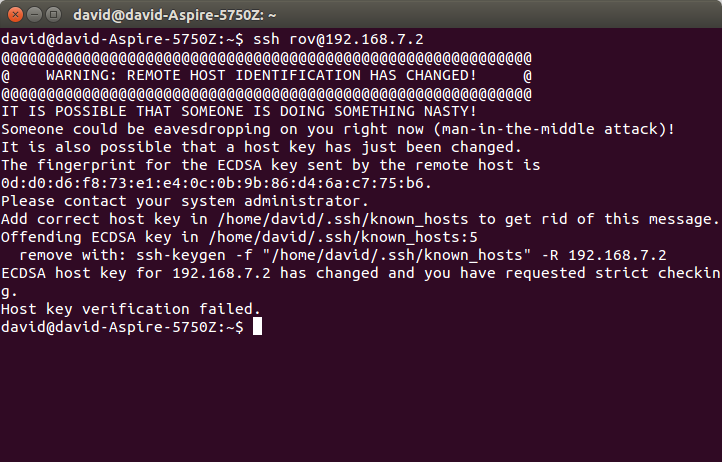
**Setup SSH:**

You configure the Beaglebone via an SSH connection.

This configuration often uses the apt-get command. The proxy connection worked fine with apt-get, however it gave errors when attempting to use it to install via npm. Because of this we recommend the following setup so that you can disable the proxy.

* Using an ethernet cable connect the Beaglebone black directly to a router
* Use a USB to mini USB cable to connect the laptop to the Beaglebone black
* ssh onto the Beaglebone black via USB (ssh rov@192.168.7.2)
* Open Cockpit in Chrome (192.168.254.1)

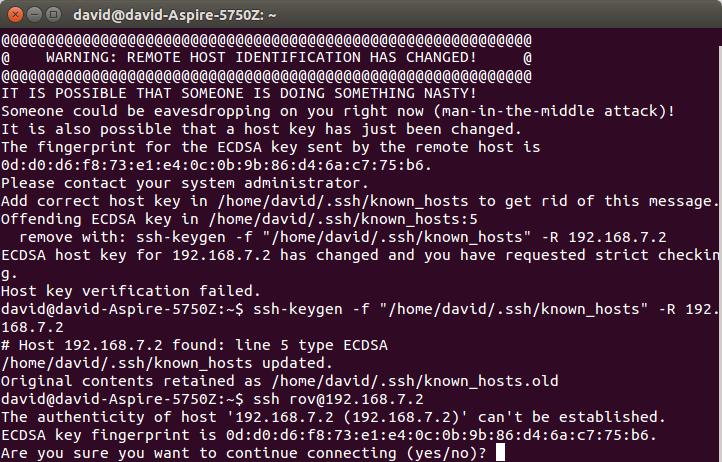
When ssh’ing onto the Beaglebone black you may get the following message, “Warning: Remote Host Identification has Changed”.



as the message mentions, you can avoid this message by entering a ssh-keygen command and repeating the ssh:

* ssh-keygen -f "/home/david/.ssh/known\_hosts" -R 192.168.7.2
* ssh rov@192.168.7.2

You may also get another warning that the authenticity of host could not be established.



type yes to continue

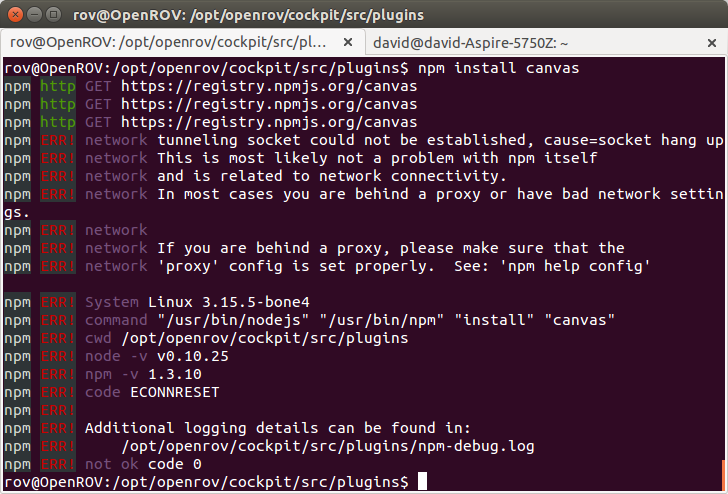
**Install dependencies (apt-get) for roslibjs:**

* sudo apt-get update
* sudo apt-get install libcairo2-dev libjpeg8-dev libpango1.0-dev libgif-dev build-essential g++
* if there are errors try again: (see screenshots)
  + sudo apt-get update
  + sudo apt-get install libcairo2-dev libjpeg8-dev libpango1.0-dev libgif-dev build-essential g++

**Install other dependencies (npm) for roslibjs:**

* sudo chown -R $USER /usr/local
* npm install canvas

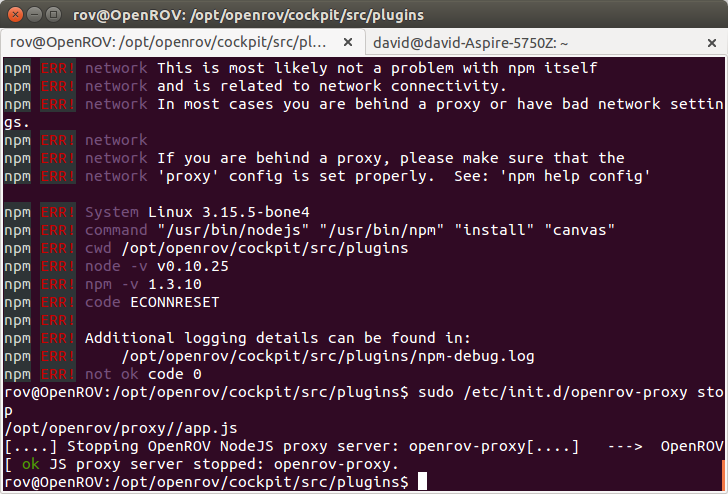
There may be errors (proxy errors) during the install:



If so, the proxy can be disabled with:

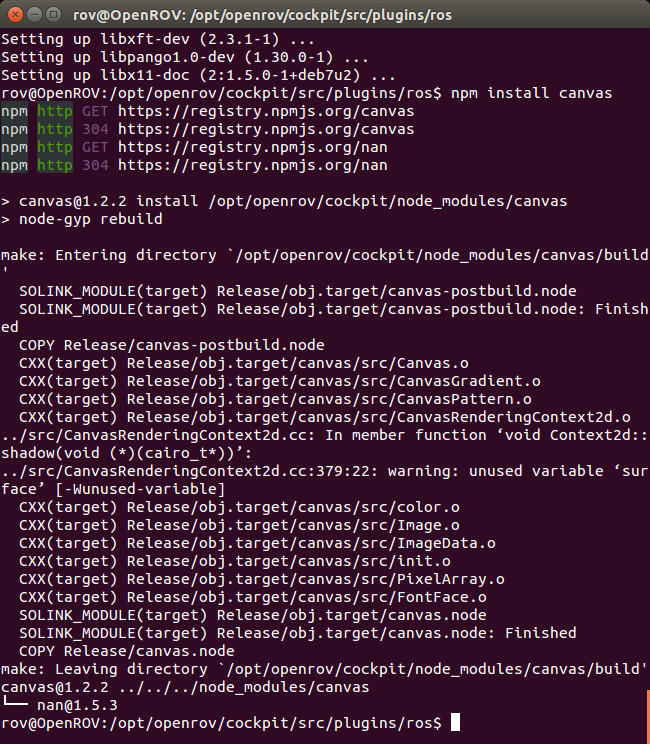
sudo /etc/init.d/openrov-proxy stop

This will take approximately 15 seconds to and show the screen shown below. Close the terminal and log on again with a new terminal session.



Retry the command: npm install canvas

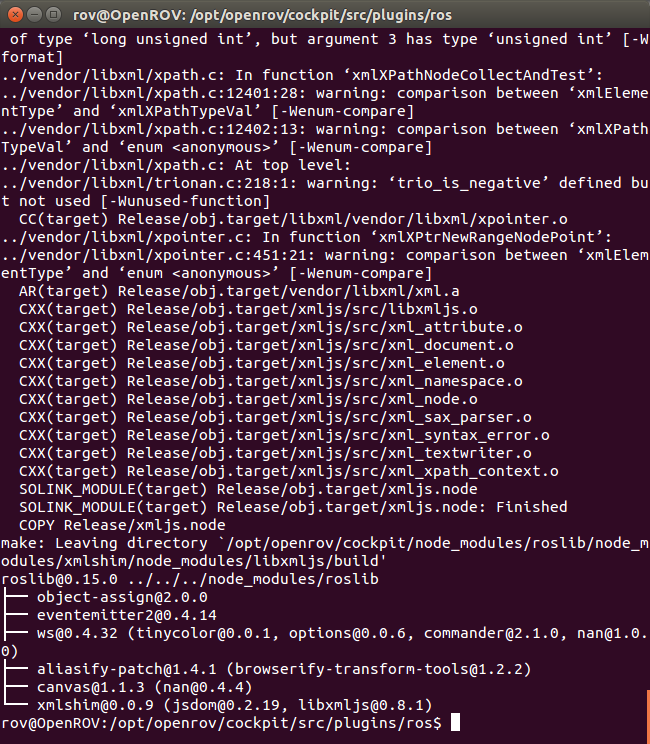
You should see the following:



**Install roslib**

The last dependency for the plugin is roslibjs, install it using: npm install roslib

This command will take several minute and the end should look like this:



**Load the ROS plugin**

Finally install the OpenROV plugin using:

* cd /opt/openrov/cockpit/src/plugins
* git clone <https://github.com/DCLane/OpenROV.git> ros

**Linux Setup**

Setting up the ROS environment for the OpenROV involves installing ROS, then installing several ROS packages and their dependencies, then installing the openrov-ros package

**Installing ROS**

* install ROS Indigo using apt-get as show here: http://wiki.ros.org/indigo/Installation/Ubuntu
  + or get a vm from here: http://nootrix.com/2014/09/ros-indigo-virtual-machine/
* create a ROS workspace (<http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment>)
* source path:
  + in .bashrc….. source /opt/ros/indigo/setup.bash
  + in .bashrc…… source ~/catkin\_ws/devel/setup.bash (according to where the working environment is setup)

**Install Rosbridge**

Install ros bridge so that the ROV can communicate using roslibjs

* sudo apt-get install ros-<rosdistro>-rosbridge-server
  + note that you must be using at least ros indigo

**Install gscam**

Install the gscam ros package and its dependencies

* sudo apt-get update && apt-get install -y \

ros-indigo-rosbridge-suite \

gstreamer0.10 \

libgstreamer-plugins-base0.10-dev \

ros-indigo-image-transport \

ros-indigo-camera-calibration-parsers \

ros-indigo-camera-info-manager \

git

* + This may take a while
* cd to the working environment setup in ‘Installing ROS’. On my machine it was ~/catkin\_ws/src
  + git clone <https://github.com/ros-drivers/gscam>
  + cd ~/catkin\_ws
  + catkin\_make

**Install OpenROV-ros**

This installs our ROS to OpenROV software:

* cd ~/catkin\_ws/src
* git clone <https://github.com/DCLane/OpenROV-ros.git> openrov
* cd ~/catkin\_ws
* catkin\_make

**Using ROS with OpenROV**

ROS is a very sophisticated and powerful system with lots of features. A full description of ROS is beyond the scope of this document. For a comprehensive tutorial on ROS see, http://wiki.ros.org/ROS/Tutorials. You should be familiar with at least the basic use of ROS to understand the rest of this section.

Communication is ROS is organized around topics. The ROS to OpenROV integration basically makes the status and workings of an ROV available as ROS topics. Once the plugin is set up on the ROV and the ROS package is set up, clients can listen to the topics that the ROV is publishing. So far these include status, navdata, temperature, pressure, and video from the webcam in several formats.

To start the ROS to OpenROV system, in separate terminal tabs launch:

* roscore
  + This starts the ROS master server
* roslaunch openrov rosbridge.launch
  + This starts the rosbridge node to communicate with the OpenROV
* roslaunch openrov webcam.launch
  + This starts a gscam node that publishes the ROVs webcam feed as ROS topics
* rqt
  + This is a ROS utility that we will use to view the video from the ROV

At this point you should see a new window for rqt (show screenshot). If the video does not automatically appear you may need to choose the image view plugin from the menu and select the correct topic, in this case /camera/visible/image. Once you have selected the correct topic rqt should remember your selection in the future.

To make sure that everything is running, open another tab in the terminal and enter:

rostopic list

This will list the topics that are being published by the ROV as well as a few default topics. You should see topics for status, navdata, temperature, pressure and various topics for video.

The topics you should see are:

* /openrov/status
  + This is sent once a second from the ROV and contains the status object as a JSON object so that all information on the ROV can be stored in a rosbag
* /openrov/navdata
  + This is sent ~20 times a second from the ROV and contains the roll, pitch, yaw, thrust, heading, and depth of the ROV.
* /openrov/temperature
  + This is sent once a second and contains the temperature reading from the ROV parsed from the status message
* /openrov/pressure
  + This is sent once a second and contains the pressure reading from the ROV parsed from the status message
* /camera/visible/image
  + This is the topic that gscam publishes the video from the ROV on. It has several subtopics with the same video with different compressions applied. ROS nodes can subscribe to it using the image\_transport package.

>show screenshot of what they should expect

You can view what is being published on a topic using rostopic echo, for example with the status topic:

rostopic echo /openrov/status

>show screenshot of status message as it should be updated each second

rostopic pub /openrov/cmd\_rate geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.5}, angular: {x: 0.0,y: 0.0,z: -0.5}}'

**Whats Next**

There are a few pieces that still need to be finished.

* Motor control, the topics that form an interface to control the ROV’s motors through ROS need to be defined but are otherwise fairly straightforward. Currently, the system listens for the /openrov/cmd\_vel topic but merely logs the messages and does not cause any actions on the ROV. This interface likely needs refactoring.
* Direct access to the raw data from the accelerometers and gyros may be possible but would involve modifying Arduino code.
* Improve the installation process and improve the documentation so that using an ROV from ROS is more straightforward.
* Add gscam as a dependency