

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.

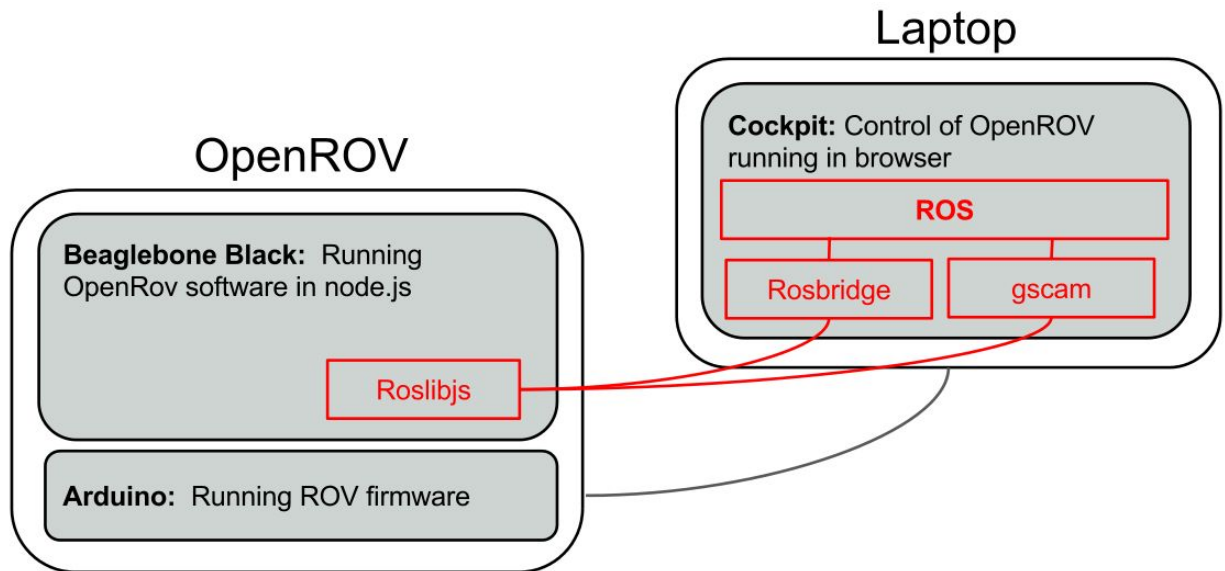
Since the ROV is tethered to a laptop and the Beaglebone Black has limited computing resources, we decided to implement ROS on the laptop instead of directly on the ROV. In this setup, the ROS Master runs on the laptop and interacts with two ROS nodes that we have installed: Rosbridge and gscam. The 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.

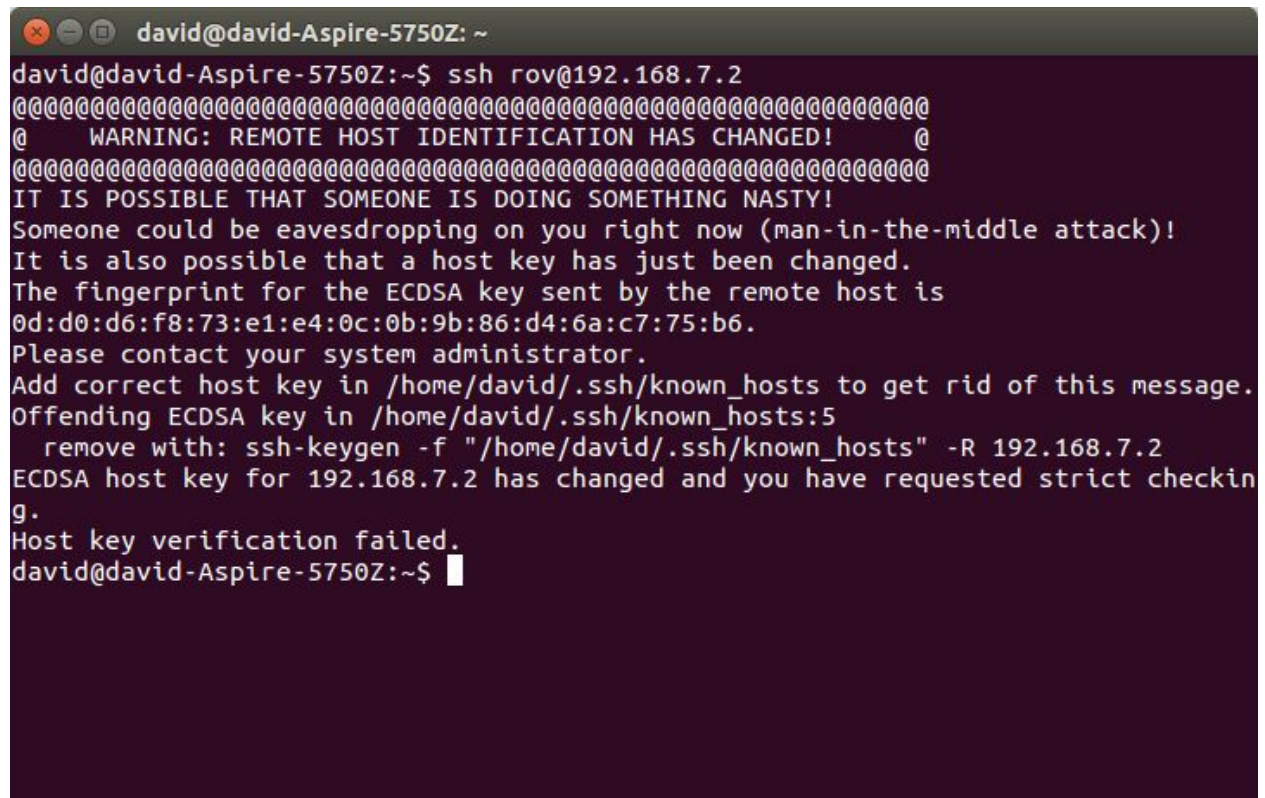
Setup SSH:

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".

A terminal window with a dark purple background and white text. The window title is "david@david-Aspire-5750Z: ~". The user has entered the command "ssh rov@192.168.7.2". The terminal displays a warning message: "WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!". It explains that it is possible someone is doing something nasty or a host key has changed. It shows the fingerprint of the ECDSA key: "0d:d0:d6:f8:73:e1:e4:0c:0b:9b:86:d4:6a:c7:75:b6". It instructs the user to contact their system administrator and to add the correct host key to the known_hosts file. It shows the offending ECDSA key in the known_hosts file: "5". It provides the command to remove the key: "ssh-keygen -f '/home/david/.ssh/known_hosts' -R 192.168.7.2". It states that the ECDSA host key for 192.168.7.2 has changed and that the user has requested strict checking. Finally, it says "Host key verification failed." and returns the prompt "david@david-Aspire-5750Z:~\$".

```
david@david-Aspire-5750Z: ~
david@david-Aspire-5750Z:~$ ssh rov@192.168.7.2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
0d:d0:d6:f8:73:e1:e4:0c:0b:9b:86:d4:6a:c7:75:b6.
Please contact your system administrator.
Add correct host key in /home/david/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/david/.ssh/known_hosts:5
  remove with: ssh-keygen -f "/home/david/.ssh/known_hosts" -R 192.168.7.2
ECDSA host key for 192.168.7.2 has changed and you have requested strict checking.
Host key verification failed.
david@david-Aspire-5750Z:~$
```

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.

```
david@david-Aspire-5750Z: ~
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
0d:d0:d6:f8:73:e1:e4:0c:0b:9b:86:d4:6a:c7:75:b6.
Please contact your system administrator.
Add correct host key in /home/david/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/david/.ssh/known_hosts:5
  remove with: ssh-keygen -f "/home/david/.ssh/known_hosts" -R 192.168.7.2
ECDSA host key for 192.168.7.2 has changed and you have requested strict checking.
Host key verification failed.
david@david-Aspire-5750Z:~$ ssh-keygen -f "/home/david/.ssh/known_hosts" -R 192.168.7.2
# Host 192.168.7.2 found: line 5 type ECDSA
/home/david/.ssh/known_hosts updated.
Original contents retained as /home/david/.ssh/known_hosts.old
david@david-Aspire-5750Z:~$ ssh rov@192.168.7.2
The authenticity of host '192.168.7.2 (192.168.7.2)' can't be established.
ECDSA key fingerprint is 0d:d0:d6:f8:73:e1:e4:0c:0b:9b:86:d4:6a:c7:75:b6.
Are you sure you want to continue connecting (yes/no)?
```

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:


```
rov@OpenROV: /opt/openrov/cockpit/src/plugins
rov@OpenROV: /opt/openrov/cockpit/src/pl... x david@david-Aspire-5750Z: ~ x
rov@OpenROV: /opt/openrov/cockpit/src/plugins$ npm install canvas
npm http GET https://registry.npmjs.org/canvas
npm http GET https://registry.npmjs.org/canvas
npm http GET https://registry.npmjs.org/canvas
npm ERR! network tunneling socket could not be established, cause=socket hang up
npm ERR! network This is most likely not a problem with npm itself
npm ERR! network and is related to network connectivity.
npm ERR! network In most cases you are behind a proxy or have bad network settin
gs.
npm ERR! network
npm ERR! network If you are behind a proxy, please make sure that the
npm ERR! network 'proxy' config is set properly. See: 'npm help config'

npm ERR! System Linux 3.15.5-bone4
npm ERR! command "/usr/bin/nodejs" "/usr/bin/npm" "install" "canvas"
npm ERR! cwd /opt/openrov/cockpit/src/plugins
npm ERR! node -v v0.10.25
npm ERR! npm -v 1.3.10
npm ERR! code ECONNRESET
npm ERR!
npm ERR! Additional logging details can be found in:
npm ERR!   /opt/openrov/cockpit/src/plugins/npm-debug.log
npm ERR! not ok code 0
rov@OpenROV: /opt/openrov/cockpit/src/plugins$
```

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.

```
rov@OpenROV: /opt/openrov/cockpit/src/plugins
rov@OpenROV: /opt/openrov/cockpit/src/pl... x david@david-Aspire-5750Z: ~ x
npm ERR! network This is most likely not a problem with npm itself
npm ERR! network and is related to network connectivity.
npm ERR! network In most cases you are behind a proxy or have bad network settin
gs.
npm ERR! network
npm ERR! network If you are behind a proxy, please make sure that the
npm ERR! network 'proxy' config is set properly. See: 'npm help config'

npm ERR! System Linux 3.15.5-bone4
npm ERR! command "/usr/bin/nodejs" "/usr/bin/npm" "install" "canvas"
npm ERR! cwd /opt/openrov/cockpit/src/plugins
npm ERR! node -v v0.10.25
npm ERR! npm -v 1.3.10
npm ERR! code ECONNRESET
npm ERR!
npm ERR! Additional logging details can be found in:
npm ERR! /opt/openrov/cockpit/src/plugins/npm-debug.log
npm ERR! not ok code 0
rov@OpenROV: /opt/openrov/cockpit/src/plugins$ sudo /etc/init.d/openrov-proxy sto
p
/opt/openrov/proxy/app.js
[....] Stopping OpenROV NodeJS proxy server: openrov-proxy[....] ---> OpenROV
[ ok JS proxy server stopped: openrov-proxy.
rov@OpenROV: /opt/openrov/cockpit/src/plugins$
```

Retry the command: `npm install canvas`

You should see the following:


```

rov@OpenROV: /opt/openrov/cockpit/src/plugins/ros
Setting up libxft-dev (2.3.1-1) ...
Setting up libpango1.0-dev (1.30.0-1) ...
Setting up libx11-doc (2:1.5.0-1+deb7u2) ...
rov@OpenROV:/opt/openrov/cockpit/src/plugins/ros$ npm install canvas
npm http GET https://registry.npmjs.org/canvas
npm http 304 https://registry.npmjs.org/canvas
npm http GET https://registry.npmjs.org/nan
npm http 304 https://registry.npmjs.org/nan

> canvas@1.2.2 install /opt/openrov/cockpit/node_modules/canvas
> node-gyp rebuild

make: Entering directory `/opt/openrov/cockpit/node_modules/canvas/build'
  SOLINK_MODULE(target) Release/obj.target/canvas-postbuild.node
  SOLINK_MODULE(target) Release/obj.target/canvas-postbuild.node: Finished
COPY Release/canvas-postbuild.node
CXX(target) Release/obj.target/canvas/src/Canvas.o
CXX(target) Release/obj.target/canvas/src/CanvasGradient.o
CXX(target) Release/obj.target/canvas/src/CanvasPattern.o
CXX(target) Release/obj.target/canvas/src/CanvasRenderingContext2d.o
../src/CanvasRenderingContext2d.cc: In member function 'void Context2d::
shadow(void (*)(cairo_t*))':
../src/CanvasRenderingContext2d.cc:379:22: warning: unused variable 'sur
face' [-Wunused-variable]
CXX(target) Release/obj.target/canvas/src/color.o
CXX(target) Release/obj.target/canvas/src/Image.o
CXX(target) Release/obj.target/canvas/src/ImageData.o
CXX(target) Release/obj.target/canvas/src/init.o
CXX(target) Release/obj.target/canvas/src/PixelArray.o
CXX(target) Release/obj.target/canvas/src/FontFace.o
  SOLINK_MODULE(target) Release/obj.target/canvas.node
  SOLINK_MODULE(target) Release/obj.target/canvas.node: Finished
COPY Release/canvas.node
make: Leaving directory `/opt/openrov/cockpit/node_modules/canvas/build'
canvas@1.2.2 ../../../../node_modules/canvas
└─ nan@1.5.3
rov@OpenROV:/opt/openrov/cockpit/src/plugins/ros$

```

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:


```

rov@OpenROV: /opt/openrov/cockpit/src/plugins/ros
of type 'long unsigned int', but argument 3 has type 'unsigned int' [-Wformat]
../vendor/libxml/xpath.c: In function 'xmlXPathNodeCollectAndTest':
../vendor/libxml/xpath.c:12401:28: warning: comparison between 'xmlElementType' and 'xmlXPathTypeVal' [-Wenum-compare]
../vendor/libxml/xpath.c:12402:13: warning: comparison between 'xmlXPathTypeVal' and 'enum <anonymous>' [-Wenum-compare]
../vendor/libxml/xpath.c: At top level:
../vendor/libxml/trionan.c:218:1: warning: 'trio_is_negative' defined but not used [-Wunused-function]
CC(target) Release/obj.target/libxml/vendor/libxml/xpointer.o
../vendor/libxml/xpointer.c: In function 'xmlXPathPtrNewRangeNodePoint':
../vendor/libxml/xpointer.c:451:21: warning: comparison between 'xmlElementType' and 'enum <anonymous>' [-Wenum-compare]
AR(target) Release/obj.target/vendor/libxml/xml.a
CXX(target) Release/obj.target/xmljs/src/libxmljs.o
CXX(target) Release/obj.target/xmljs/src/xml_attribute.o
CXX(target) Release/obj.target/xmljs/src/xml_document.o
CXX(target) Release/obj.target/xmljs/src/xml_element.o
CXX(target) Release/obj.target/xmljs/src/xml_namespace.o
CXX(target) Release/obj.target/xmljs/src/xml_node.o
CXX(target) Release/obj.target/xmljs/src/xml_sax_parser.o
CXX(target) Release/obj.target/xmljs/src/xml_syntax_error.o
CXX(target) Release/obj.target/xmljs/src/xml_textwriter.o
CXX(target) Release/obj.target/xmljs/src/xml_xpath_context.o
SOLINK_MODULE(target) Release/obj.target/xmljs.node
SOLINK_MODULE(target) Release/obj.target/xmljs.node: Finished
COPY Release/xmljs.node
make: Leaving directory `/opt/openrov/cockpit/node_modules/roslib/node_modules/xmlshim/node_modules/libxmljs/build'
roslib@0.15.0 ..../node_modules/roslib
├─ object-assign@2.0.0
├─ eventemitter2@0.4.14
├─ ws@0.4.32 (tinycolor@0.0.1, options@0.0.6, commander@2.1.0, nan@1.0.0)
├─ aliasify-patch@1.4.1 (browserify-transform-tools@1.2.2)
├─ canvas@1.1.3 (nan@0.4.4)
├─ xmlshim@0.0.9 (jsdom@0.2.19, libxmljs@0.8.1)
rov@OpenROV:/opt/openrov/cockpit/src/plugins/ros$

```

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-<rostdistro>-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 in 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