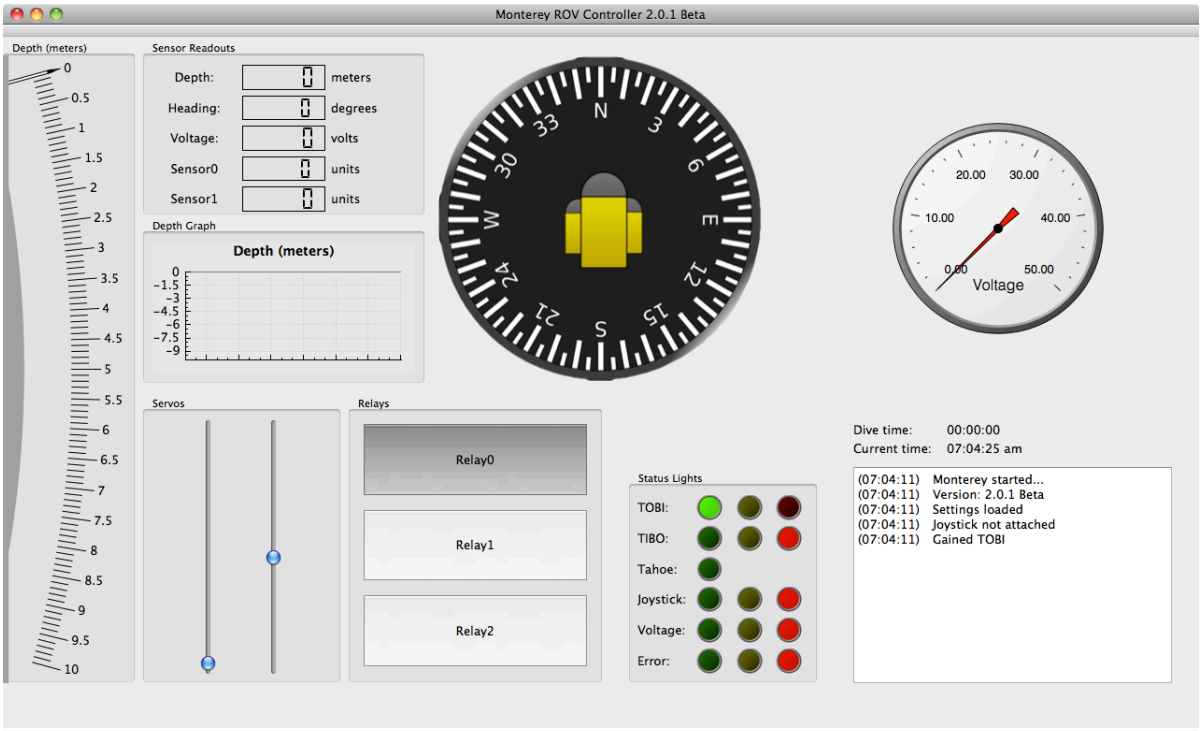
ROV-Suite Documentation

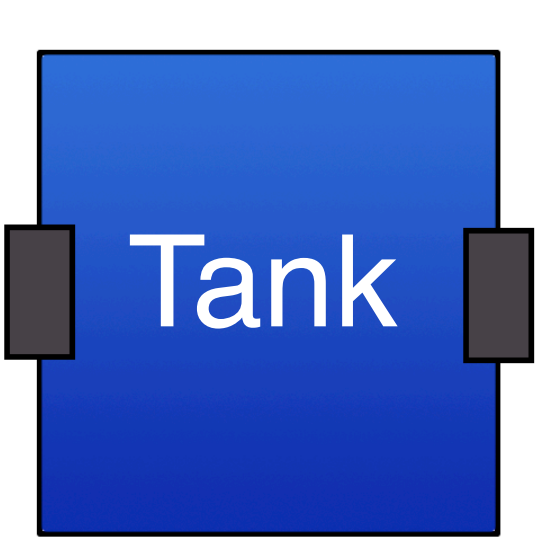
Revision 1.0
June 16th, 2012
Written By: Chris Konstad



Introduction

ROV-Suite is an open sourced ROV piloting and control application developed in Qt (C++). It is cross platform, support Windows, OSX and Linux. It was originally developed during Q4 of 2011, but then it was put on hold while the developer worked on an ROV controller for his robotics team. Development continued on ROV-Suite in the Spring of 2012. With the lessons learned from making the ROV controller for his team, Chris Konstad (the original developer) was able to write a cleaner code base for Monterey. Since it's origins, Monterey has changed greatly and the concepts behind ROV-Suite have also evolved. What started out as a monolithic application has now been pared down to a simple control interface with multiple windows. Support for more features ("Tahoe", the portable GUI, etc) have also been added since the conception of ROV-Suite.

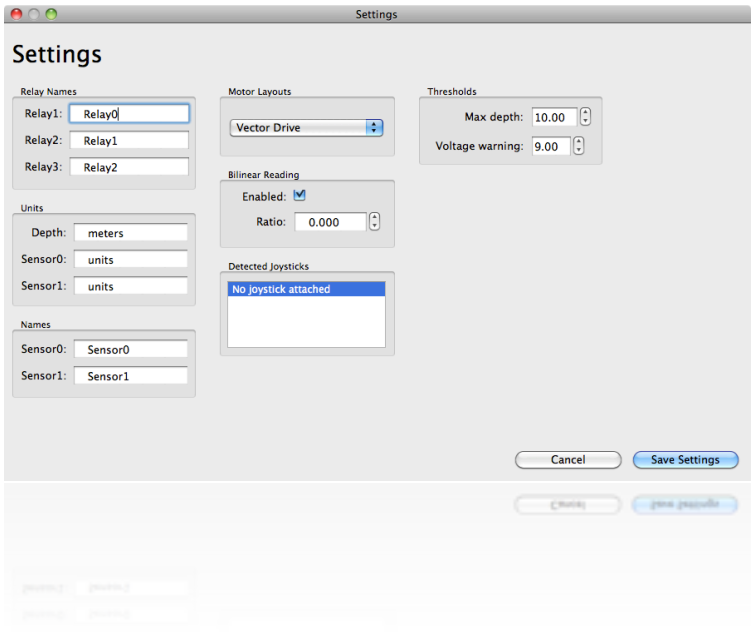
There are a few key underlying concepts that have remained constant in ROV-Suite's lifetime. First and foremost, ROV-Suite has been developed to fit your ROV. It has support for three relays, two servos and five sensors right out of the box. You can change the functionality of several parts of the GUI (units involved, ranges, etc) without even knowing any programming. Monterey also supports two main motor layouts, "vector" and "tank" drive (vertical motors not pictured).



Finally, Monterey's code is made to be as easily modifiable as possible. It was written with many QLists and "foreach" loops to make it simple to add sensors, motors, relays, and servos to it.

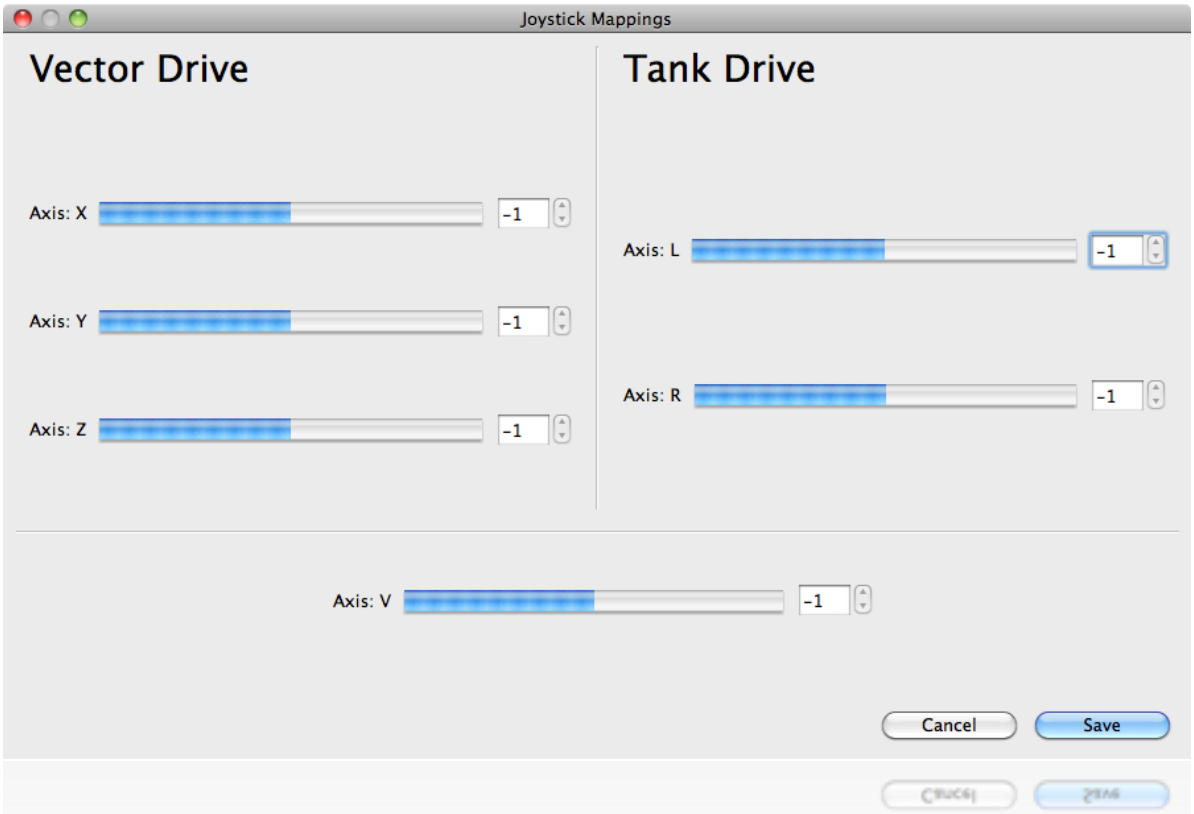
How To Use

One of the main forces driving Monterey's development was usability. Originally, in Monterey (v1.0), you could modify pretty much every value straight from the Settings GUI. In theory, that doesn't sound like that could be a problem. The issue is that it makes the Settings GUI too complex, hard to navigate, and it opens up the system too much. Monterey v2 is different in that



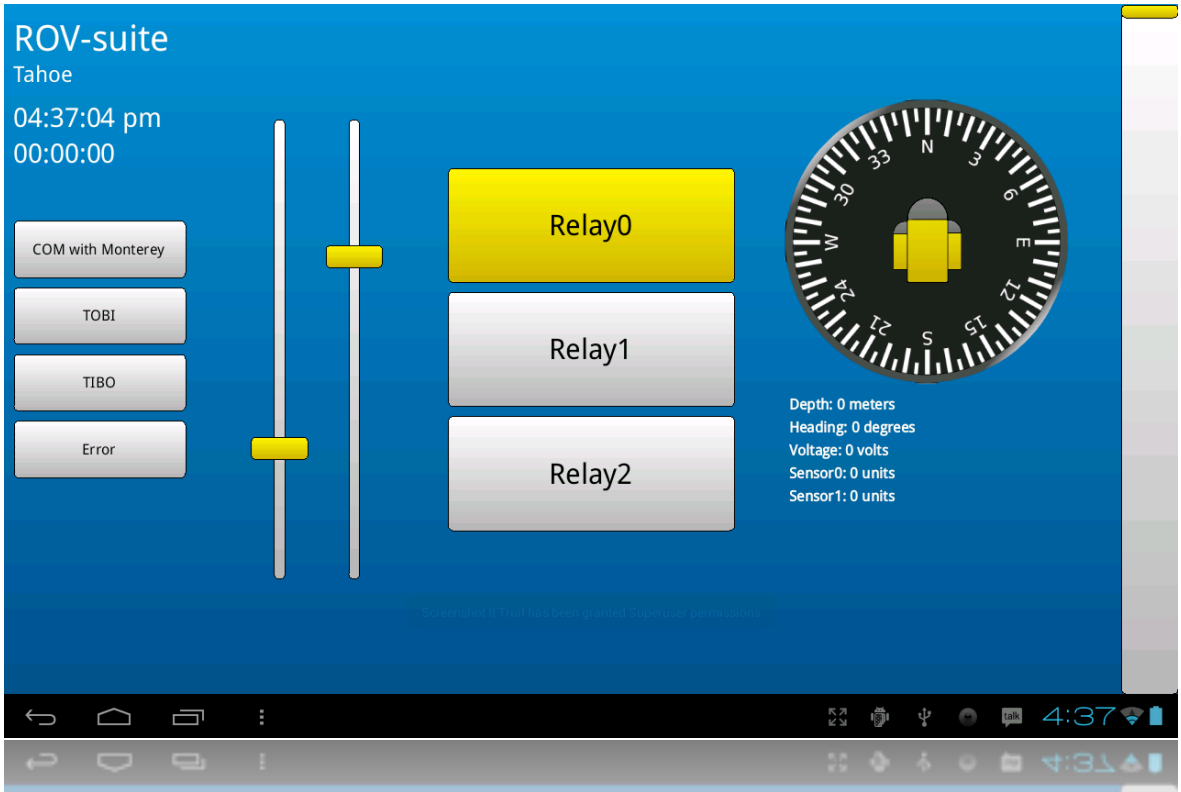
regard. The Settings GUI still lets the user change many (arguably all) commonly changed settings. This simplifies the code, the GUI and the operations of Monterey.

Monterey v2 couldn't be much easier to use. After flashing a compatible bottomside software package to your ROV and changing some settings, simply plug your tether's ethernet line into your computer's ethernet jack (and it's recommended that you turn Wi-Fi off). Then start up Monterey. Finally, when the ROV is powered on, Monterey will automatically establish communication with it. Any HIDs (human interface device) that you want to use (joysticks, "Tahoe", etc) must be set up as well. To set up the joystick properly, open up the Joystick Mappings window. From there, you can choose which axis modify which values (x, y, rotation, vertical for vector and left, right and vertical for tank)



Monterey is made to fit your ROV and your setup. When you modify the settings, Monterey saves them to a file called "settings.ini." On OSX, it is stored in the Monterey application bundle. On Windows it is stored in the folder that Monterey (and corresponding DLLs) are. You can copy that *.ini file between computers to copy over all of Monterey's settings. Monterey automatically syncs settings (relay names, sensor units and names, etc) between itself and any devices running the Tahoe portable GUI.

Tahoe



Tahoe is the portable GUI (written in QML) that is able to be compiled for any OS that supports Qt (Android, iOS, OSX, Windows, Linux, etc). It pulls the settings from Monterey automatically whenever it runs. It has indicators (on the left) that show the if it is communicating with Monterey, if Monterey has both TOBI and TIBO (top-out-bottom-in and top-in-bottom-out) COMs, and if there are any errors. Eventually, Tahoe and Monterey will sync states of the servos and relays in both directions. As of Monterey 2.0.1 Beta and Tahoe 1.0, however, Tahoe will overwrite Monterey's commands before they are sent to the ROV.

Bottomside

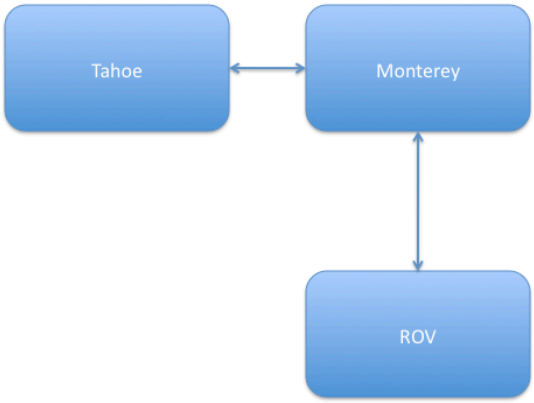
There's not too much to say about how to use the bottomside. You can either write your own (discussed in more detail below) or use the one provided. In both cases, you must make sure that your hardware is properly coded for (sensors, ESCs, etc) and wired up. Obviously, if your wiring is wrong, then the software won't work (let me tell from 3 years on an underwater robotics team, wiring is more often than not the issue). With the provided bottomside software, simply flash it to the Arduino and wire it up. Once the ROV is powered on, it will automatically connect to Monterey (over an Ethernet cable).

The Underlying Code

LIke stated above, Monterey has been written from the very beginning to be easily modifiable. After all, it's your ROV, so the software should conform to you (not the other way around). While several aspects are modifiable from the Settings GUI, some changes may require you to modify the source and recompile. You can find the source from the ROV-Suite SourceForge project page (http://www.sourceforge.net/p/rov-suite). It is stored in a Git repo. On OSX, SourceTree is a great free Git client. On Windows, TortoiseGit is a great free Git client. If you're on Linux, I'm sure you already have Git installed. Also, you must have Qt installed to recompile Monterey. I recommend that you just download the Qt SDK. You can then compile from either a GUI or the command line. If compiling from the command line, just go to the Monterey directory that houses "Monterey.pro". From there, type "qmake Monterey.pro", "make" and "make install" and you're done! You will have just built your new version of Monterey.

Monterey is developed as MVC (model, view, controller) based application. The model in this case is the QROV object. That houses the states of all relays, servos, motors and sensors as well as their units, names, etc. If you want to add more sensors, servos, relays or motors, that is where you'll add them. Next, the QROVController object is the object that handles the math, inputs, and creates the UDP packets that are sent to the ROV. It can, in theory, run entirely by itself. In fact, it runs in a thread separate from the GUI.

Monterey communicates with Tahoe (version 1.0) and the bottomside software over UDP (user-datagram-protocol). UDP is used because it is robust and reliable. Because there is no hand-shaking involved and it is a lossy protocol, if packets are randomly lost, there is no visible impact on performance. Later on, Tahoe might be switched from a UDP connection to a TCP/IP connection to allow for event-based communication.

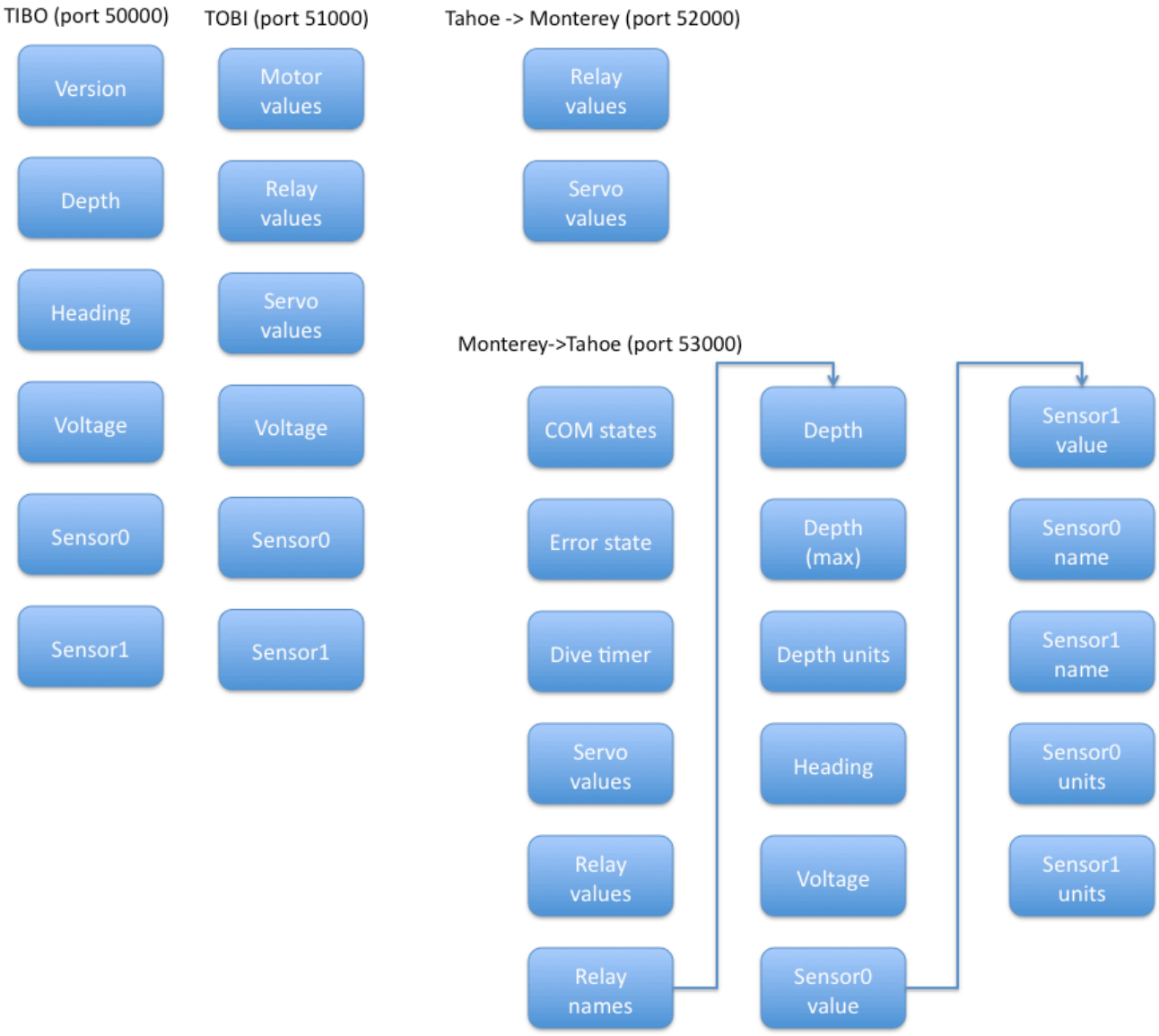


The ROV control network based on UDP

Packet Layouts

The communication (at the time of this writing) between Tahoe/Monterey, and Monterey/ROV is all broadcasted UDP based. By using broadcasting, the IP addresses of each device do not matter and it is easy to use ROV-Suite's NetworkingTest to catch

packets for easy debugging. This helps create a robust reliable connection. The user will really only experience a loss of communication if the physical connection (wired or Wi-Fi) is broken or if the computer (or ROV, etc) freezes. Below, you will see diagrams of the UDP packets and the ports that they are sent to (when using UDP, you specify which port you send to and the OS chooses the port that it send from by itself because the port of origin does not really matter).



Contact Us

If you have any feedback, questions, concerns or ideas about the program, project, documentation or anything else, please let us know! We'd love to hear from you! You can find out project page at SourceForge at http://www.sourceforge.net/p/rov-suite. Our development blog can be found at http://chrisstechblog.blogspot.com. There, you will find articles talking about the latest developments of the code and you can comment about our great (and not so great) future features.

If you run into *any* bugs, *please* file a ticket at our SourceForge project page! Someone else has more than likely run into the same bug (or will soon), so the sooner you let us know about it the sooner we can fix it!

Thank you for your participation in this project and happy ROVing!

Chris Konstad
ROV-Suite