Individual Lab Report
Weekly Progress

Brian Boyle
Team B: Team Space Jockey
Teammates: Nathaniel Chapman, Ardya Dipta Nandaviri, Songjie Zhong

ILR02
October 17, 2013

1. Individual Progress

My primary role within this lab was to develop a Graphical User Interface (GUI) for the Space Jockey robot. Secondarily, I also did some research into ROS and what other software would be compatible with it, and as a team we agreed to use it for our system. Because of this change in planned software architecture, I had to use a different software package for GUI development that would work in the ROS environment. This package, PyQT, comes with its own design program for creating GUI layouts, then exports a configuration file that can be used in a variety of software environments. This is not only more flexible than Tkinter, but also much easier to design with. The end product has four “tabs” that can be selected to display distinct sub-interfaces.

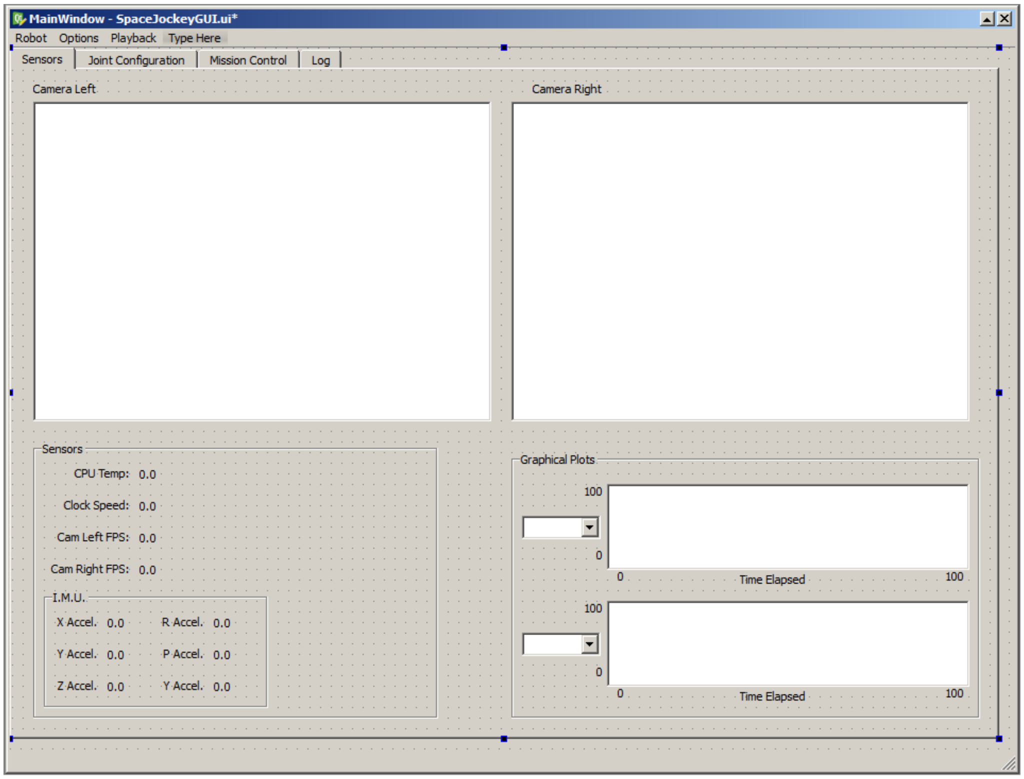
The first tab (See Figure 1) displays camera and sensor feedback. Both left and right cameras are shown in separate canvas elements. Sensors are listed in the lower-left pane with their current live value, and in some cases are grouped (As in the I.M.U.) values. There is ample space remaining for additional sensor readouts, and the ones shown will not necessarily appear in the final version but are shown mainly as demonstrative examples. The lower-right pane includes space to graphically plot sensor outputs over time, with a drop down menu that can select between the various sensors. Future versions of this GUI may allow the overlaying of multiple sensor time series in a single plot.

The second tab (See Figure 2) is used to monitor joint configurations as well as compose, manage, and execute custom configurations for testing or manual teleoperation. A wireframe simulation of the robot will be shown in the upper left quadrant that by default will show its current state. To its right are three radio buttons which allow the user to also show a “target” configuration, which would be a queued configuration to which the robot is currently moving, or a “composed” configuration, which is one that the user is creating using the below interface. In the lower left area, readouts of joint positions for each individual leg are shown. In the lower right, the user can design configurations joint-by-joint then save, load, or send it to the robot as a command. The user is also able to copy the current configuration of the robot as a starting point to composing a new one, zero all the values with a single click, or issue an emergency “Halt” command to the robot which stalls all motors.

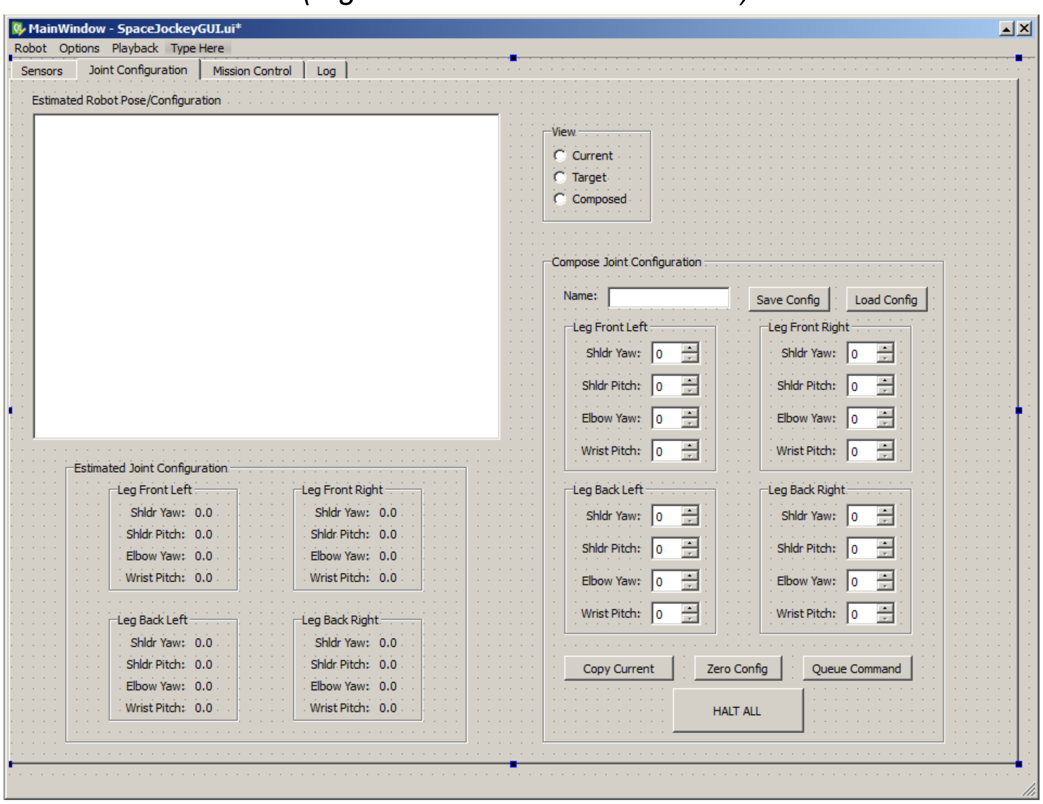
The third tab (See Figure 3) will serve as the main interface during missions. The upper left pane will display a map of the environment, waypoints, and a representation of the robot within it. The upper right pane will display either the left camera, right camera, a disparity map between the two cameras, or a height map, selectable using the drop-down menu directly below the pane. The bottom-right area displays some key mission information (Duration, distance travelled, and battery remaining), and a list of waypoints with buttons to add, remove, and rearrange their ordering.

The fourth and final tab (See Figure 4) is devoted to text logging from the robot and the software operating it. The menus on the right allow the user to save, load, and export log files, as well as select via checkbox the categories of messages to display. These messages will scroll by live in the left pane as the robot executes.

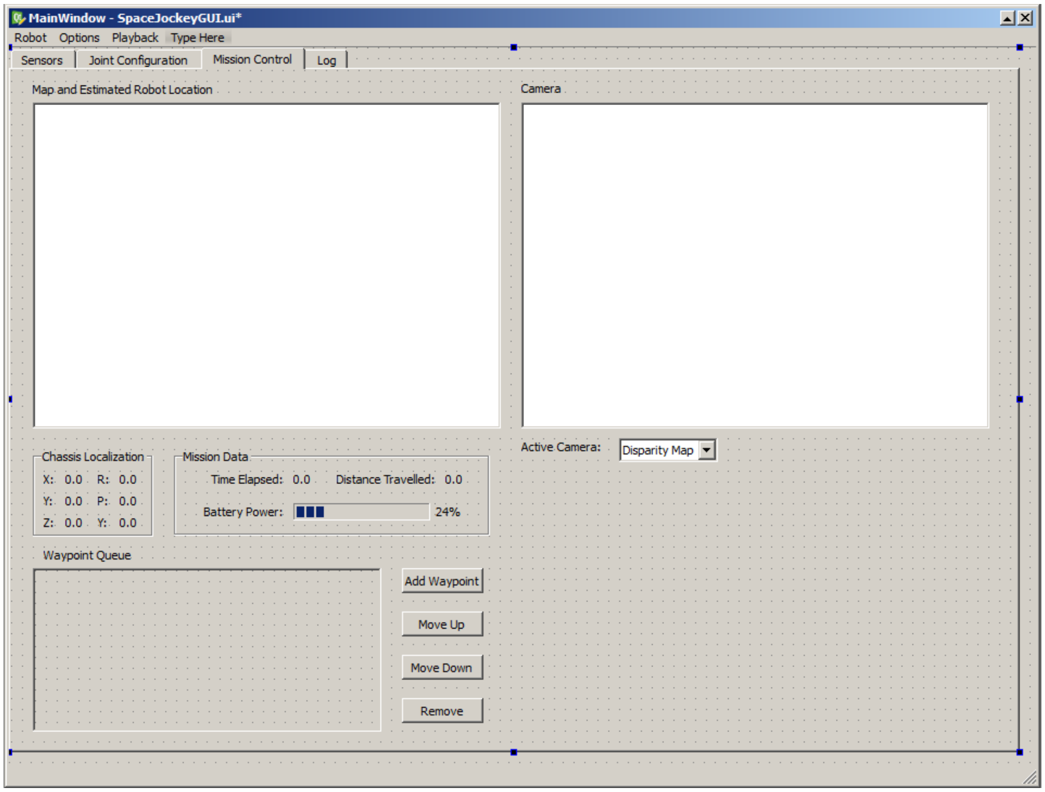
Finally, a “Playback” menu at the top of the window contains options to record and play back GUI states. Ben Wasserman informed me that there’s a ROS module which can record the entirety of messages passed between nodes, so hopefully such a feature can be leveraged to allow us to review system tests through the GUI as though we were performing them again live. Another possibility may be using such a feature to record GUI events as a way of composing unit and functional tests for the system.



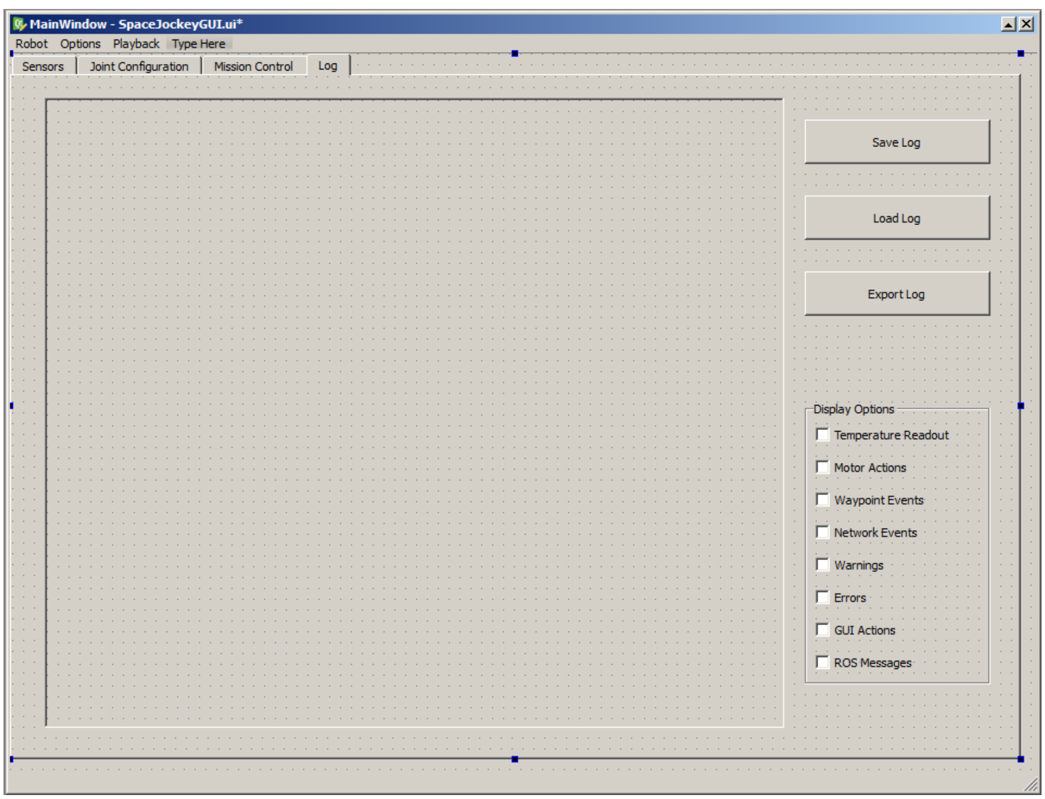
(Figure 1: "Sensors" tab of GUI)



(Figure 2: "Joint Configuration" tab of GUI)



(Figure 3: "Mission Control" tab of GUI)



(Figure 4: "Log" tab of GUI)

2. Challenges/Issues

By switching to the QT package for GUI's, the design and implementation of the GUI was much less challenging than in ILR when I was using Tkinter. However, the scope of this GUI was much broader, as it was developed with both our validation tests and final demo in mind. As such, it was necessary to consider all the functionalities we desired, then devise interfaces to accomplish them, then arrange those interfaces in an intuitive and efficient way. This process took place on paper for the first few drafts then the transfer to QT was relatively painless after learning my way around the user interface. Drafting on paper certainly saved a lot of time, as did the QT design interface.

3. Team Work

While I built the GUI, my teammates accomplished some major work towards our physical mechanisms and electronics. Dipta performed a trade study on various motors for use in the project and also developed our specification and design for a power supply system. With the trade study completed, Nate was able to order some servo motors, a microcontroller, motor shield, and other assorted parts that will go into further refining our bench squid and connecting it to the GUI.

With motors selected, Songjie was then able to import models for them into solidworks and model some a leg concept. I worked with him beforehand to make sure he knew how to export these models for use by the Solidoodle 3D printer. Finally, Songjie and Dipta have had success locating material for dry-adhesive “Gecko” feet, a material that we can use to cast them like one would plaster.

4. Future Work

For our next progress review, we'll be undergoing a substantial redesign of our robot, switching from a legged mobility solution to one more resembling an inchworm. This solution was stumbled upon while we were brainstorming with Dimi, and I look forward to fleshing it out. Nate, Songjie, and Dipta will be focusing on developing electrical and mechanical systems for this new design, while I'll adopt our current GUI to be applicable to it.

I also plan to continue researching ROS, and conduct a trade study to determine exactly which distribution will be most useful for our particular robot. I plan to also work with Nate while he integrates and tests our recently-ordered components so that I'll be already familiar with the robot-side architecture when the time comes to integrate it with the GUI, hopefully avoiding some of the hecticness of our integration process in ILR 01.