Project Proposal

A Web Interface for Jackal Navigation, Mapping, and Control

Daniel Randle

2022-10-20

1 Background

UAF purchased the Clearpath Jackal and Grizzly as assets to aid in mining emergency operations. Pogo Mine was interested in exploration and mapping capabilities and Clearpath provided plaftorms which were, in many ways, turn key for the application. Several graduate students worked alongside ACUASI to further develop the platforms, and to integrate UAS systems.

The Jackal and Husky provide many untapped capabilities, largely due to inconvenient user interface and out-of-date software. The platforms utilize ROS for communication and control; Clearpath has released updated Linux images running newer versions of ROS and new navigation/mapping nodes since the platforms were acquired. The platforms also include base station tripod wifi extenders, along with long range wifi antennas on the robots in order to provide an extended wifi network range. These extenders are no longer operational, and the robot wifi modules are no longer configured correctly to create a single AdHoc wifi network.

2 Summary

Without moderate ROS knowledge, the Jackal and Husky cannot be used to their full capabilities. I propose creating a web interface which allows anyone with a smart phone to connect and control the Jackal/Husky through their web browser. Since an Android/IOS app would be beneficial, and since I am most proficient coding in C/C++, I will use the MIT licensed Urho3D library to create the application. Urho3D allows using the same C/C++ to build for multiple targets including Android, IOS, and Web (through Emscripten). Along with the app, I will create a custom ROS node with a simple navigation algorithm utilizing the navigation stack to demonstrate using a custom navigation node through the app.

Very often these platforms are used as demos to children/young adults that may be interested in the EE program. In addition to providing opportunities for future students to implement new custom navigation behavior more easily, this project would allow the demonstration of the platforms' full capabilities by using the app to show the current mapping data in real time.

3 Outline

This project can be broken into several interdependent tasks. Each of these tasks will be documented in the final project report.

3.1 Platform Update

Both the Jackal and Husky need OS and ROS updates to Ubuntu 20.04/ROS Noetic respectively for multiple reasons:

1. ROS Indigo (which is currently installed on Jackal/Husky) requires Ubuntu 13.10 or 14.04. This means communicating and controlling ROS nodes on the robot platforms require computers with these versions of Ubuntu (which may only be possible through VM since it is so old).
2. New navigation nodes available from Clearpath require ROS Noetic (or Melodic) which requires Ubuntu 20.04. In particular, gmapping isn't supported for ROS Indigo and so to use it, it would need to be built from source.
3. Coding new ROS nodes in the ROS Indigo/Ubuntu 14.04, even if a computer with old hardware made it possible, would be painful. All community help and new community ROS nodes are written using newer ROS versions.

Updating the OS requires installing the Clearpath specific Linux distribution, setting up ROS Noetic, and setting up all parameters to bring the platform sensors online.

3.2 Base Station WiFi Restoration

Both base stations need to be restored, and the wifi AdHoc network re-configured.

1. Base station bullet routers need firmware updates and reconfiguration as AdHoc network extenders
2. Bullet routers in Husky and Jackal need reconfiguration as AdHoc network extenders
3. Network adaptors in Husky and Jackal CPU need to be configured to connect to network correctly

3.3 Node.js Web Server

Although through Emscripten, Urho3D takes source C/C++ and compiles it into html linked to javascript and WebAssembly files, the resulting page must be served from the Husky/Jackal over wifi to clients. To do this, I will utilize Node.js to create a simple web server on port 80 with routes set up to correctly serve the pages.

Since the resulting html page, which provides the user interface for navigation, control, and mapping, runs in the client's browser there must also be a mechanism to exchange data to/from the Jackal/Husky in full duplex. Without this, the client must constantly poll for updates. To accomplish this, a WebSocket will be opened and listen on another port on the Node.js server. This WebSocket will be used to provide the client with all needed navigation and mapping data, and relay client navigation/control commands to ROS using Node.js ROS libraries.

3.4 Web/Android App

The app will provide the user interface to command/control and view mapping data. The app should do the following:

1. Allow picking between manual driving and automated navigation
2. For manual driving, provide a joystick and customizable top speed
3. For automated navigation, allow picking between different navigation modes/algorithms
4. Allow drawing markers and rectangles on the map in order to set navigation goals for the automated navigation algorithms
5. Show current location of robot with respect to current scan data, gmapping SLAM map, and cost map (available as overlays)
6. Allow auto registration of new navigation ROS nodes without needing to recompile source html

3.5 Custom Navigation ROS Node

A simple navigation node using cost map, user marker input, and possibly A* or D* pathfinding to get to the marker avoiding obstacles along the way.

3.6 Simulation

It is inconvenient to physically drive the robots in order to test the navigation and app. To aid future students that may wish to develop the platform further, I will go through the steps of setting up simulation using Gazebo and running the web application to control the simulated robot.