**Robos – Robot Operating System Concept Report**

Andrew Wood

**The Problem:**

This project is focused on delivering a robust and extensible platform on which Robot code can execute in a safe, controlled manner. Robots have lots of different components ranging from simple such as sensor drivers, to very complex such as local planners. A proper Operating System for a robot needs to handle and synchronize all components running in the system. Traditionally, the problem of synchronizing components has fallen on the component programmers to implement, resulting in somewhat buggy, but more importantly, unmaintainable code. If robotics code needs to be changed; a component doesn’t work or a component is being phased out, large chunks of the system need to be updated to properly reflect the change.

To solve this problem, ROS (Robot Operating System) was invented. It solves this problem by taking a more abstract approach; the downfall of the previous method of coding involved components themselves passing information to other components that required on it (a simple example being a sensor would directly pass its sensor information to a planner that performed logic based on that information) and instead ROS itself passed along the information. All a component has to do is “publish” their information as a Message, and ROS would route that Message to any component who needed the information to execute. ROS can be viewed as a “routing system,” it is not a true Operating System (it is not bootable), but acts like an Operating System in that it manages all components running on the robot.

**Project Concept:**

Robos is a similar product as ROS. It is a Robot Operating System that controls the execution of all components running on the robot, as well as manages the information flow between components. The difference is that Robos is much smaller than ROS, ROS is complicated for new users, and takes up a lot of memory to store. For small robots that do not have a lot of memory, using ROS is very costly. Robos is also designed as a cross platform program, meaning that it can run on any commercial Operating System (Windows, Linux, OSX, etc.) while ROS is only supported by Linux. This design choice is significant in helping programmers unfamiliar with Linux develop robot code, as becoming adjusted to a Linux Operating System to use ROS can be a significant hurdle.

Robos is built as a C++ dynamic library that sits atop an Asynchronous Framework. It provides a small build script for client code to compile with, and will provide as a byproduct of the client compilation process, a small executable that launches Robos customized to client specification. This project includes finishing and optimizing the current handwritten Asynchronous Library, creating a type system for client code to populate, a Framework for message distribution, an easily scalable Database solution for client components that Robos controls, and appropriate code for method invocation of client code. If time permits, a GUI representation of the running Robos system and potential resource checking will be created. All in all, Robos is designed to be a lightweight version of ROS that provides a development friendly environment that is easily customizable, while being robust enough to guarantee execution correctness of client programs.