XeroBoard Design

# Introduction

This application is an attempt to move the computer-based applications that communicate with the robot via network tables to a more modern development framework that is extensible by robotics teams.

# Design Goals

* Base the system on the NT4 publish/subscribe model for network tables.
* Have a single application support multiple ‘applets’ that interact with network tables. This leads to a single network tables connection to the robot and the network table data can be shared among applets.
* Use modern front-end technologies for creating the user interface for various applets.
* Ensure it is easy for teams to create their own applets.
* Create an applet that is a “shuffleboard” like interface that supports the current shuffleboard semantics and provides an easy-to-use extension method for creating custom widgets.

# Design

The results of looking at these design goals and formulating a design is shown in the diagram below. Several alternatives were considered including running a http server on the robot for provide information directly to a browser, but in the end the decision was to limit the connection to the robot to the network tables connection only. No specific additional robot code should be required.

The XeroBoard application is an electron application. It is assumed that the reader has a passing understanding of the electronjs architecture. See the web site <https://www.electronjs.org/> for more information on electron.

All electronjs applications consist of a main process that runs as a nodejs application and one or more browser processes that looks mostly like the Chrome browser would look on the desktop. The only exception to this is that the electronjs framework provides for a narrow and tightly controlled communications interface between the main process and the browser processes.

Applets consist of a frontend and a backend. The frontend uses normal web-based UI technologies (HTML, CSS, JS) to provide visualization. The backend runs in the electron main process and provides support infrastructure for the frontend.

The main process manages the connection to the robot network tables over an NT4 network tables connection. Each applet has access to the NT4 client and can subscribe or publish as desired. The resultant data is stored in a central and shared network table data repository. This implies more data may be present than any single applet requested, and care must be taken when accessing the data.

The main process creates an HTTP server on the local loop back address (127.0.0.1) on a dynamically assigned port that the browser processes and the server shares. Applets are launched by loading a specific HTML page via this HTTP server.

The main process creates a browser window that runs a special browser application that manages electron-tabs (<https://github.com/brrd/electron-tabs>). There is an IPC channel that provides control information about what applets should be launched. This control application iterates over each of the applets to be launched, creates a tab with a WebView for that applet, and loads the specific HTML page for the applet.

The browser portion of an applet can have any number of CSS, JS, and HTML files. These files are part of the applet and are served by the http server created in the main process. The main process portion of an applet can provide any number of REST API services to support the browser portion of the applet as well. The browser portion of an applet may also access network table data via the Network Table Query API. This is an API that provides IPC based access to the network table data in the main process.

