University of Nevada, Reno

MachinView

A thesis submitted in partial fulfillment
of the requirements for the degree of

COMPUTER SCIENCE AND ENGINEERING, BACHELOR OF SCIENCE

by

JOSH CURTIS

SERGIU DASCALU, PhD. Thesis Advisor

May, 2016

UNIVERSITY OF NEVADA RENO

THE HONORS PROGRAM

We recommend that the thesis
prepared under our supervision by

JOSH CURTIS

entitled

MachinView

be accepted in partial fulfillment of the
requirements for the degree of

[NAME OF DEGREE, e.g., BACHELOR OF ARTS, PSYCHOLOGY]

_____

Sergiu Dascalu, Ph.D., Thesis Advisor

_____

Tamara Valentine, Ph.D., Director, Honors Program

May, 2016

1. Abstract

3D printers require custom software to operate. Hobbyists who build their own 3D printers must create or edit their own software. Learning to write code for 3D printers can be a barrier for many people who want to create their own 3D printers. The goal of this project is to design and implement a graphical user interface (GUI) that will allow hobbyists to easily create custom software to run and manage their 3D printers. The prototype operates on a BeagleBone running Snappy Ubuntu Core. The application is a modified version of MachineKit with a web application, written in Clojurescript, for interfacing with the BeagleBone remotely. The group is advised by Dr. Richard Kelley and Jake Mestre.

2. Introduction

The main goals of this project are to provide a user-friendly method of specifying the parameters and editing configuration files of custom designed 3D printers. Additionally, the project aims to provide provide monitoring software that communicates over a local network for 3D printers. The target audience for this project are 3D printer hobbyists that may not have the necessary experience programming to set up their printers.

To encourage more advanced users to use this application, another goal of this project is to provide a method for sending commands remotely to the 3D printer controller. By providing this functionality, the project will be beneficial for controlling the printer as well. The desired end result of this project is to reduce the startup time between physically constructing the 3D printer and using the 3D printer.

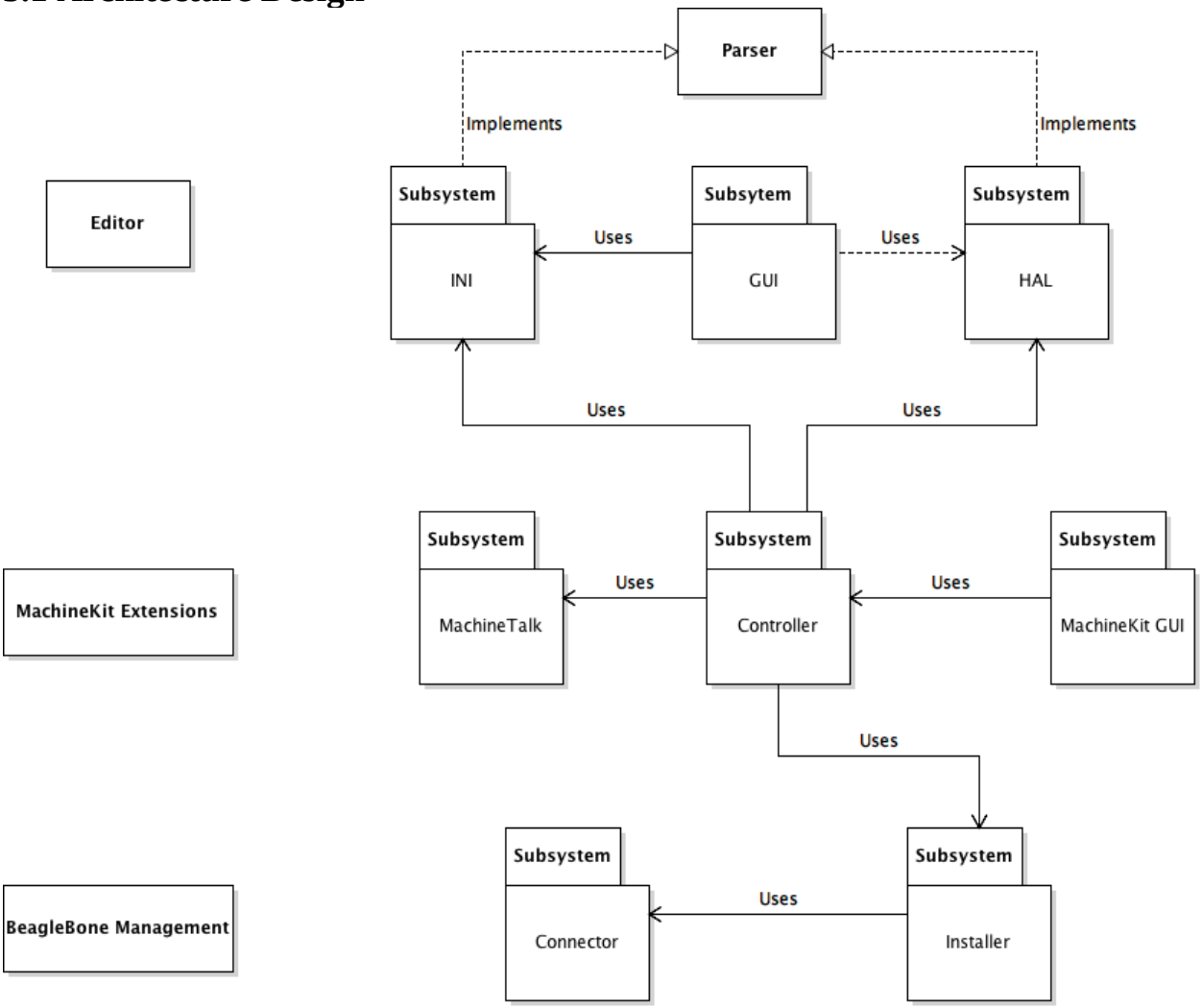
Since the previous report, the team has continued to experiment with the source of MachineKit, MachineFace, and MachineTalk. Additionally, the team added more validation to the configuration editing component of the project. The most significant change is the team's decision to move away from modifying MachineKit internally and to implement the project using MachineTalk. This change will allow the team to add features that involve communicating with the 3D printer controller easier than editing and compiling a custom version of MachineKit.

The change to focus on working with MachineTalk was the result of a significant development challenge. MachineKit is structured in a highly coupled manner that prevents any easy modification to the code that would allow the team to continue making progress. Additionally, due to midterms and work schedules, the team was not able to devote as much time to working on the project since the last project report. To address this, the team plans on spending available time during Spring Break to make more progress on the communication component.

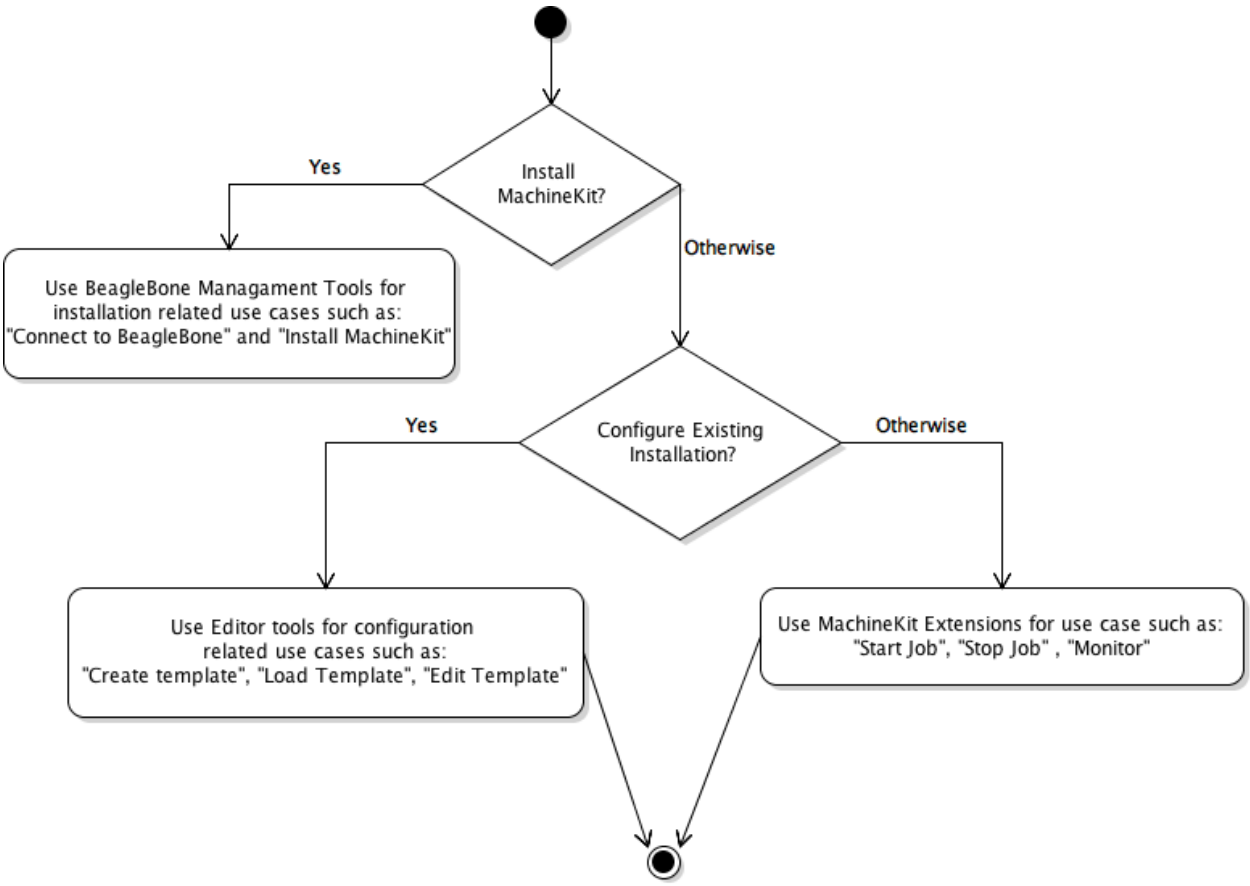
3. Design Model

The design is aimed to be implemented in Clojure/Clojurescript. Clojure is a dialect of lisp. It is predominantly a functional programming language with a rich set of immutable data structures. Clojurescript is compilable to JavaScript and is well suited to making GUIs.

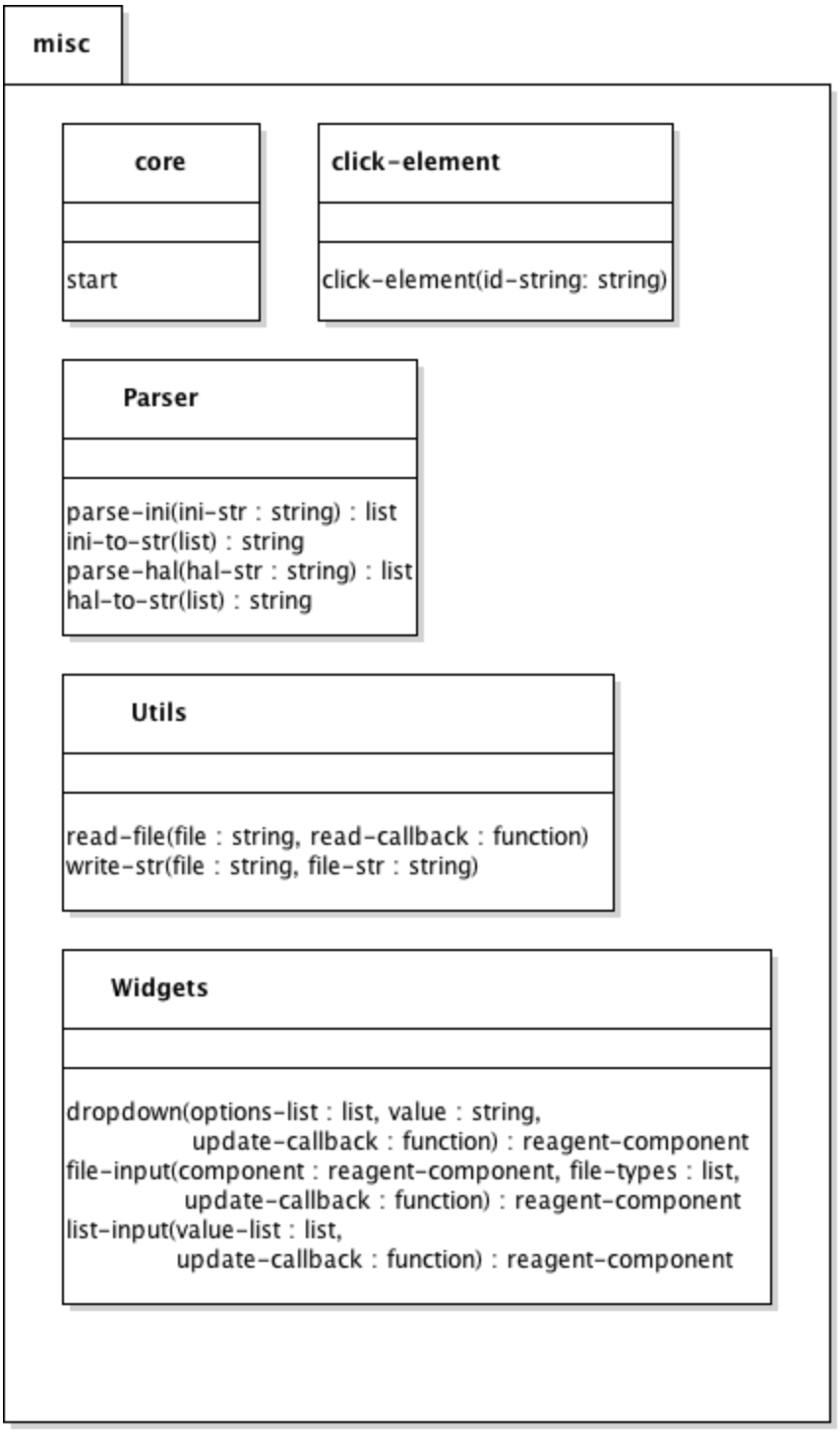
3.1 Architecture Design

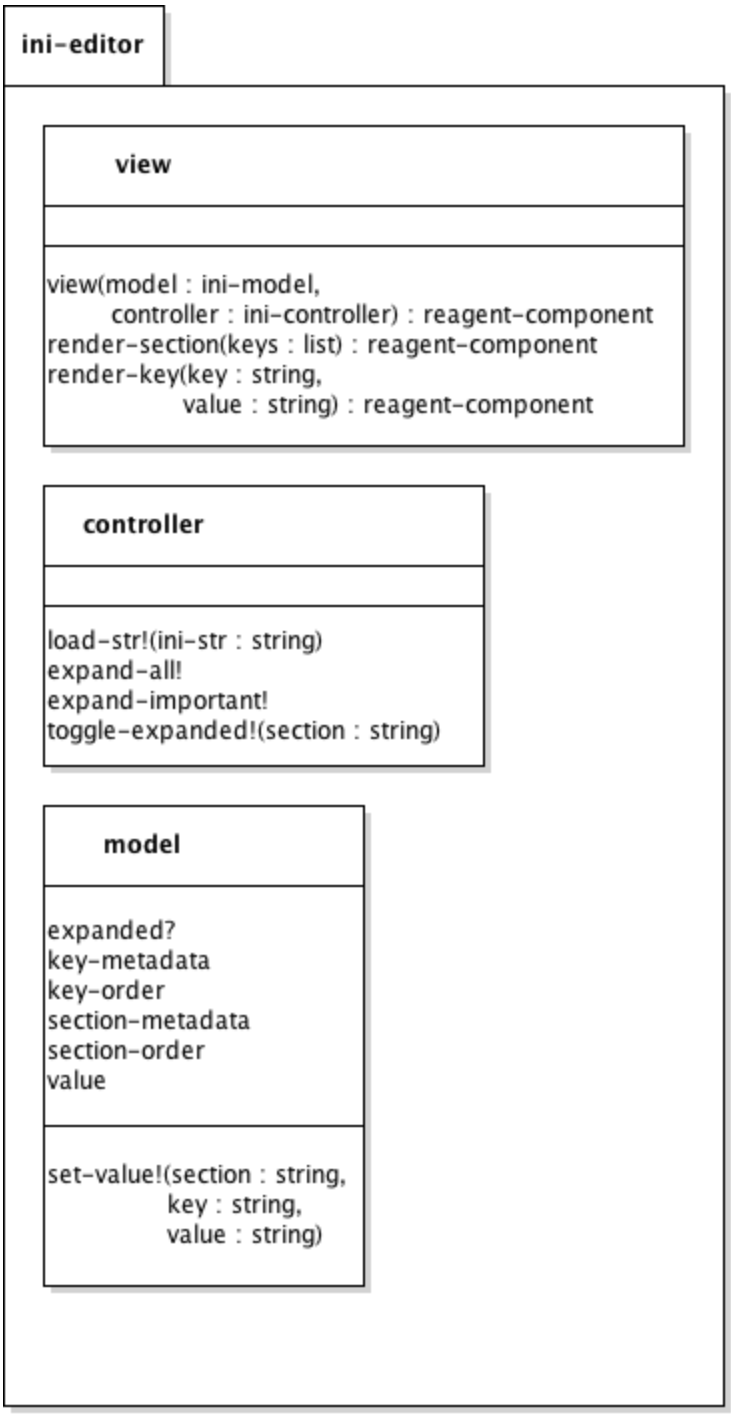


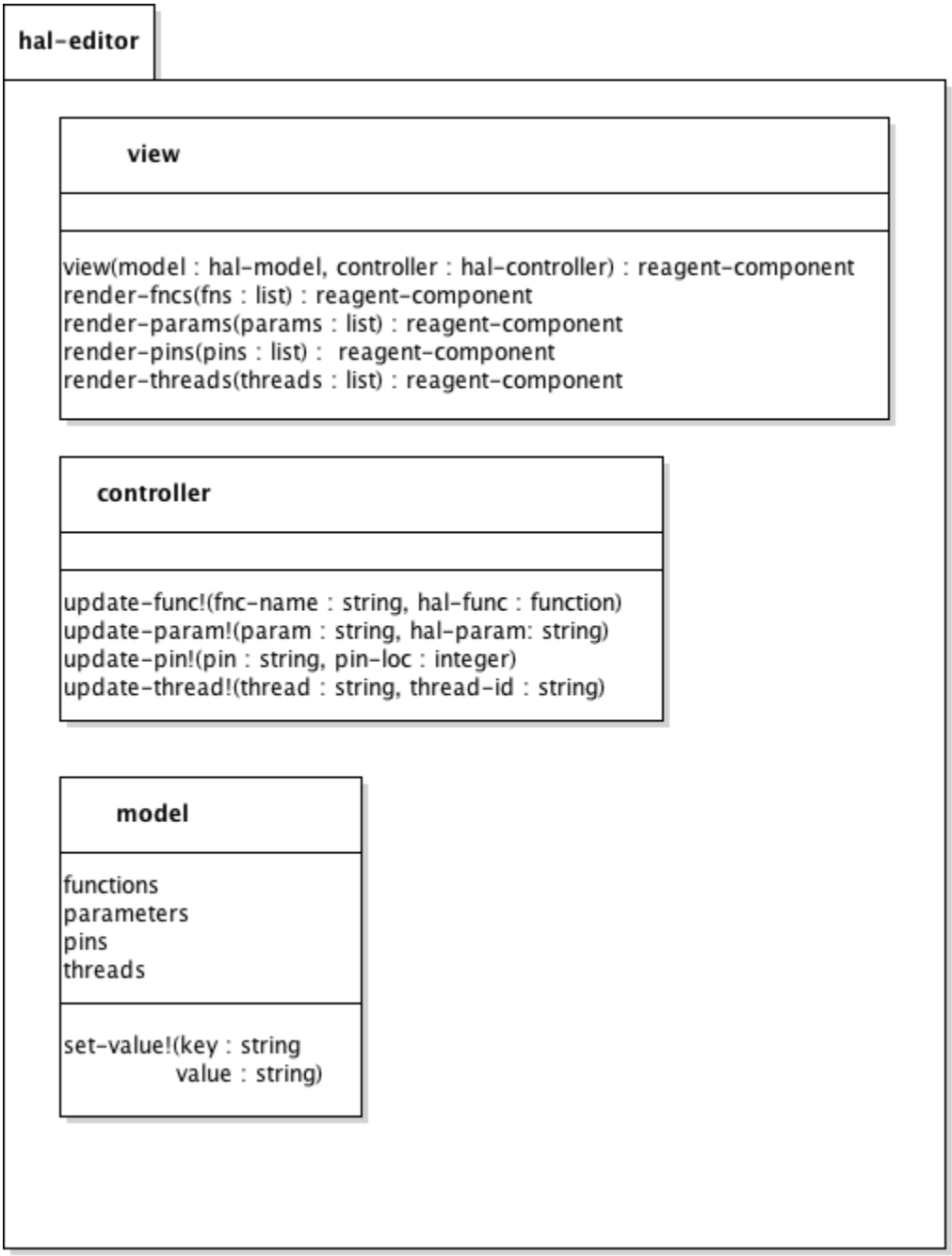
- The state diagram below shows that the user will interact with one of three components when using our system depending on their intended use case.

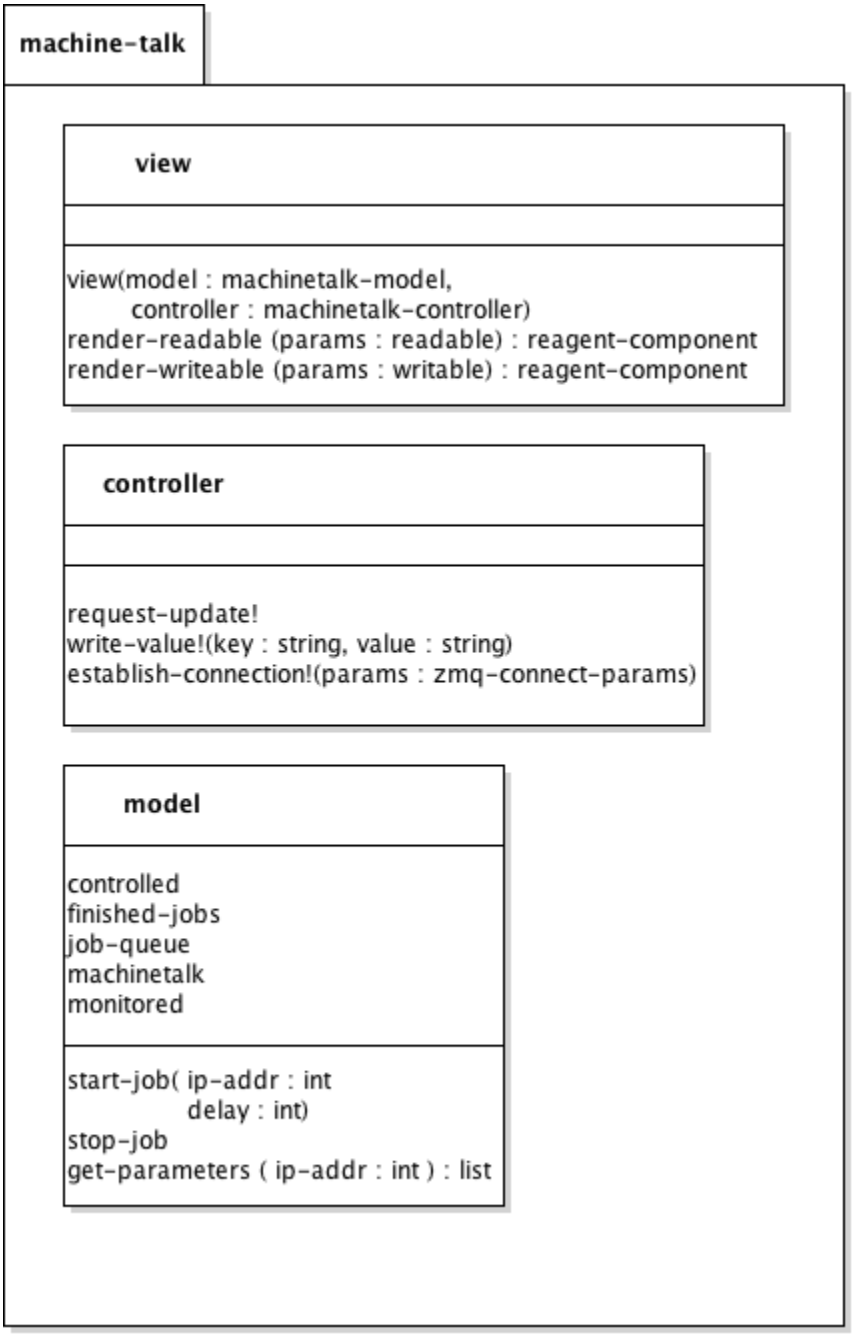


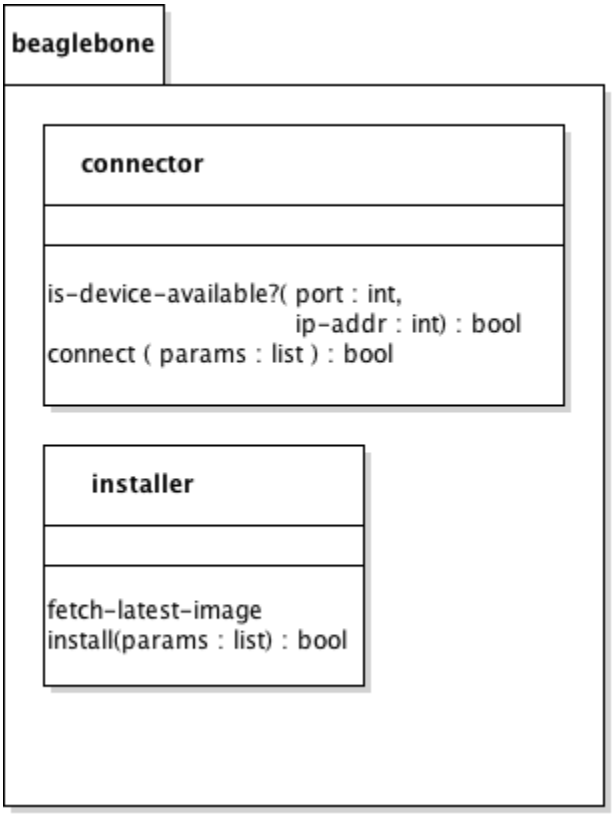
3.2 Class Diagrams


misc
core
start
click-element
click-element(id-string: string)
Parser
parse-ini(ini-str : string) : list
ini-to-str(list) : string
parse-hal(hal-str : string) : list
hal-to-str(list) : string
Utils
read-file(file : string, read-callback : function)
write-str(file : string, file-str : string)
Widgets
dropdown(options-list : list, value : string,
update-callback : function) : reagent-component
file-input(component : reagent-component, file-types : list,
update-callback : function) : reagent-component
list-input(value-list : list,
update-callback : function) : reagent-component


ini-editor
view
view(model : ini-model,
controller : ini-controller) : reagent-component
render-section(keys : list) : reagent-component
render-key(key : string,
value : string) : reagent-component
controller
load-str!(ini-str : string)
expand-all!
expand-important!
toggle-expanded!(section : string)
model
expanded?
key-metadata
key-order
section-metadata
section-order
value
set-value!(section : string,
key : string,
value : string)


hal-editor
view
view(model : hal-model, controller : hal-controller) : reagent-component
render-fncs(fns : list) : reagent-component
render-params(params : list) : reagent-component
render-pins(pins : list) : reagent-component
render-threads(threads : list) : reagent-component
controller
update-func!(fnc-name : string, hal-func : function)
update-param!(param : string, hal-param: string)
update-pin!(pin : string, pin-loc : integer)
update-thread!(thread : string, thread-id : string)
model
functions
parameters
pins
threads
set-value!(key : string
value : string)


machine-talk
view
view(model : machinetalk-model,
controller : machinetalk-controller)
render-readable (params : readable) : reagent-component
render-writeable (params : writable) : reagent-component
controller
request-update!
write-value!(key : string, value : string)
establish-connection!(params : zmq-connect-params)
model
controlled
finished-jobs
job-queue
machinetalk
monitored
start-job(ip-addr : int
delay : int)
stop-job
get-parameters (ip-addr : int) : list



3.3 Program Units

Name: dropdown	Module: widgets
Description: Renders a dropdown reagent-component with an easy to use interface. Preconditions: None Postconditions: update-callback will be called if the user selects an option in the dropdown. Inputs: options-list, value, update-callback Outputs: reagent-component Exceptions: None Author: Josh	

Name: file-input	Module: widgets
Description: Renders a component that is capable of querying the user for a file, once clicked. Preconditions: None Postconditions: update-callback will be called if the user choses to upload a file. Inputs: component, file-types, update-callback Outputs: reagent-component Exceptions: None Author: Will	

Name: list-input	Module: widgets
Description: Renders a component that is capable of managing a list. Has functionality such as removing, editing, and adding elements. Preconditions: None Postconditions: update-callback will be called if the user modifies the data. Inputs: value-list, update-callback Outputs: reagent-component Exceptions: None Author: Will	

Name: parse-ini	Module: parser
Description: Parses an ini string into data that is compatible with ini-editor.model. Preconditions: None Postconditions: None Inputs: string Outputs: :key-metadata, :key-order, :section-metadata, :section-order, :value. Exceptions: parse-error Author: Will	

Name: ini-to-str	Module: parser
Description: Turns ini data into a string that is compatible with MachineKit. Preconditions: None Postconditions: None Inputs: key-metadata, key-order, section-metadata, section-order, value Outputs: string Exceptions: parse-error Author: Will	

Name: parse-hal	Module: parser
Description: Parses a hal string into data that is compatible with hal-editor.model. Preconditions: None Postconditions: None Inputs: string Outputs: :functions :parameters :pins :threads Exceptions: parse-error Author: Will	

Name: hal-to-str	Module: parser
Description: Turns hal data into a string that is compatible with MachineKit. Preconditions: None Postconditions: None Inputs: functions, parameters, pins, threads Outputs: string Exceptions: parse-error Author: Will	

Name: read-file	Module: utils
Description: Allows for reading of files from disk. The file is read asynchronously. Preconditions: None Postconditions: file is read to string and passed to read-callback when ready, Inputs: js-file, read-callback Outputs: None Exceptions: file-error Author: Shub	

Name: write-str	Module: utils
Description: Allows for writing of a string to disk. This is done asynchronously. Preconditions: None Postconditions: string is written to file Inputs: js-file, string Outputs: None Exceptions: file-error Author: Shub	

Name: click-element	Module: click-element
Description: Clicks a DOM element. This hack is required for some widgets. Preconditions: None Postconditions: DOM element with the specified id is clicked. Inputs: id-string Outputs: None Exceptions: element-not-found-error Author: Shub	

Name: start	Module: core
Description: Initially renders the reagent component onto the DOM element. This builds the initial page the user will see when starting the application. Preconditions: None Postcondition: Application is rendered onto the page. Input: None Output: None Exceptions: None Author: Josh	

Name: view	Module: ini-editor.view
Description: Creates a renderable reagent component that displays the contents of the ini that is being edited. Preconditions: None Postconditions: None Input: ini-editor.model, ini-editor.controller Output: reagent-component Exceptions: None Author: Josh	

Name: render-section	Module: ini-editor.view
Description: Renders a reagent component which represents a section in the ini configuration. It is used by ini-editor.view/view. Preconditions: None Postconditions: None Input: section-name, section-values-hashmap, section-metadata, key-order, key-metadata Output: reagent-component Exceptions: None Author: Will	

Name: render-key	Module: ini-editor.view
Description: Renders a reagent component which represents a key value pair in the ini configuration. It is used by ini-editor.view/render-section. Preconditions: None Postconditions: None Input: key-name, key-value, metadata Output: reagent-component Exceptions: None Author: Will	

Name: load-str!	Module: ini-editor.controller
Description: Reset the INI model and load it from the provided string. If the provided string is not a valid configuration, then an exception is thrown and the model is left empty. Preconditions: None Postconditions: All variables in ini-editor/model.cljs may be updated if successful. Input: ini-string Output: None Exceptions: parse-error Author: Will	

Name: expand-all!	Module: ini-editor.controller
Description: All ini sections are expanded. Preconditions: None Postconditions: ini-editor.model/expanded? Will include all sections. Input: None Output: None Exceptions: None Author: Josh	

Name: expand-important!	Module: ini-editor.controller
Description: All ini sections that are not tagged as unimportant are expanded while the remaining are collapsed. Preconditions: None Postconditions: None Input: None Output: None Exceptions: None Author: Josh	

Name: toggle-expanded!	Module: ini-editor.controller
Description: The specified section will be expanded if it is collapsed, or collapsed if it is expanded. Preconditions: None Postconditions: ini-editor.model/expanded? is updated.	

Input: section-string
Output: None
Exceptions: None
Author: Josh

Name: set-value!	Module: ini-editor.model

Description: Checks if the provided section key value combination is valid. If it is, then the data is updated.
Preconditions: None
Postconditions: ini-editor.model/value is updates if the combination is valid. A warning is set if failed.
Input: section-string, key-string, new-value
Output: None
Exceptions: None
Author: Josh

Name: view	Module: hal-editor.view

Description: Creates a renderable reagent component that displays the contents of the hal that is being edited.
Preconditions: None
Postconditions: None
Input: hal-editor.model, hal-editor.controller
Output: reagent-component
Exceptions: None
Author: Josh

Name: render-funcs	Module: hal-editor.view

Description: Renders a graphical interface to view and edit functions from the hal that is being edited.
Preconditions: None
Postconditions: None
Input: funcs
Output: reagent-component
Exceptions: None
Author: Josh

Name: render-params	Module: hal-editor.view

Description: Renders a graphical interface to view and edit parameters from the hal that is being edited.
Preconditions: None
Postconditions: None

Input: parameters
Output: reagent-component
Exceptions: None
Author: Josh

Name: render-pins	Module: hal-editor.view

Description: Renders a graphical interface to view and edit pins from the hal that is being edited.
Preconditions: None
Postconditions: None
Input: pins
Output: reagent-component
Exceptions: None
Author: Shub

Name: render-threads	Module: hal-editor.view

Description: Renders a graphical interface to view and edit threads from the hal that is being edited.
Preconditions: None
Postconditions: None
Inputs: hal-threads
Output: reagent-component
Exceptions: None
Author: Shub

Name: update-func!	Module: hal-editor.controller

Description: Adds or updates the hal function if the new configuration is valid, or issues warning if it is invalid.
Preconditions: None
Postconditions: None
Inputs: function-name, hal-function
Outputs: None
Exceptions: func-error if the function is invalid.
Author: Josh

Name: update-param!	Module: hal-editor.controller

Description: Adds or updates the hal parameter if the new configuration is valid, or issues warning if it is invalid.
Preconditions: None
Postconditions: None

Inputs: param-name, hal-param
Outputs: None
Exceptions: param-error if the function is invalid.
Author: Will

Name: update-pin!	Module: hal-editor.controller
Description: Adds or updates the hal pin if the new configuration is valid, or issues warning if it is invalid. Preconditions: None Postconditions: None Inputs: pin-name, hal-pin Outputs: None Exceptions: pin-error if the function is invalid. Author: Will	

Name: update-thread!	Module: hal-editor.controller
Description: Adds or updates the hal thread if the new configuration is valid, or issues warning if it is invalid. Preconditions: None Postconditions: None Inputs: thread-name, hal-thread Outputs: None Exceptions: thread-error if the function is invalid. Author: Will	

Name: view	Module: machine-talk.view
Description: Creates a renderable reagent component that displays the state of the remote 3D printer, if the application is connected. Preconditions: None Postconditions: None Input: machinetalk.model, machinetalk.controller Output: reagent-component Exceptions: None Author: Shub	

Name: render-readable	Module: machine-talk.view
Description: Displays readable values from the printer setup. Preconditions: None Postconditions: None Input: readable-params Output: reagent-component Exceptions: None Author: Shub	

Name: render-writeable	Module: machine-talk.view
Description: Displays writeable values from the printer setup. They are also able to be modified remotely. Preconditions: None Postconditions: None Input: writeable-params Output: reagent-component Exceptions: None Author: Shub	

Name: request-update!	Module: machine-talk.controller
Description: Requests an update from the remote printer. This call happens asynchronously, so there will be a delay between when it is called and when it completes. Preconditions: Application is connected to printer Postconditions: machinetalk.model variables will be updated once the information is retrieved. Input: None Output: None Exceptions: not-connected-error Author: Shub	

Name: write-value!	Module: machine-talk.controller
Description: Requests to write a value in the remote 3D printer, if it is valid. If it is not valid, then a warning will be issued. Preconditions: Application is connected to printer Postconditions: machinetalk.model/controlled will be updated Input: parameter-name, parameter-value Output: None Exception: wrong-parameter-error Author: Shub	

Name: establish-connection!	Module: machine-talk.controller
Description: Attempts to connect with a remote printer. Preconditions: None Postconditions: machinetalk.model variables will be modified if the connection is successful. Input: zmq-connect-parameters Output: None Exception: connection-failed-error Author: Shub	

4. Data Design

For GUI components, the state is located in its respective model module. For example, the following files contain app state information, ini-editor/model.cljs, hal-editor/model.cljs, machinetalk-interface/model.cljs. The other files provide immutable data transformations, with the exception of utils.cljs, which provides asynchronous io through callbacks.

ini-editor/model.cljs		
Variable	Description	Type
expanded?	The sections which are expanded.	ClojureSet #{section-string}
key-metadata	Holds information which helps for viewing and editing. Some fields include :type, :options, and :verification-func.	ClojureHashMap section-string -> key-string -> ClojureHashMap
key-order	The order in which the keys of a section should be displayed.	ClojureHashMap section-string -> [key-string]
section-metadata	Holds information which helps for viewing or editing. Some fields include :important? And :comment	ClojureHashMap section-string -> ClojureHashMap
section-order	The order in which the sections should be displayed.	ClojureVector [section-string]
value	Holds the parameters assigned.	ClojureHashMap section-string -> key-string -> string or [string]

hal-editor/model.cljs		
Variable	Description	Type
functions	Functions are a ClojureHashMap with properties such as :pin, :value, :dir, :owner, and :action	ClojureHashMap func-string -> ClojureHashMap
parameters	Parameters map to ini keys. The sections include :section, :key, and :metadata	ClojureHashMap param-string -> ClojureHashMap
pins	Mappings between pin names and their physical properties. Some of the properties include :owner, :type, :direction, :value, :namespace, and :name.	ClojureHashMap pin-string -> ClojureHashMap
threads	List of threads. A thread has the properties :name :period :fp? :time and :max-time.	ClojureList `(ClojureHashMap)

machinetalk/model.cljs		
Variable	Description	Type
controlled	ClojureHashMap of components that are being controlled in MachineKit. The properties include :type, :value, :nickname, :name, :range, and :verification-func.	ClojureHashMap component-name -> ClojureHashMap
finished-jobs	A list of jobs that have been finished. The properties include :name, :time-queued, :run-time, and :job-status.	ClojureList ‘(ClojureHashMap)
job-queue	A queue of jobs that have not been sent. They are sent sequentially as soon as it is verified that the printer is not busy. The properties of a job include the strings :gcode, :name, and the time :time-queued.	ClojureQueue ClojureQueue(ClojureHashMap).
machinetalk	Handle that is used to communicate with the 3D printer.	ZmqContext
monitored	ClojureHashMap of components that are being monitored from MachineKit. The properties include :type, :value, :nickname, and :name.	ClojureHashMap component-name -> ClojureHashMap

8. Glossary of Terms

3D printer	A 3D printer refers to any machine capable of generating a 3D object given some 3D model.
Beaglebone Black	A cheap single board computer that is more powerful than the Raspberry Pi. It has significantly more IO and processing power.
Bootstrap	An open source CSS library published by Twitter. It provides CSS classes that greatly improve the visual design of web pages over basic HTML
Cartesian Printer	A type of 3D printer. The extruder can move along a grid the X, Y, and Z planes.
CNC Machine	A computerized numerical control machine describes machines that are given instructions via computer rather than humans.
Delta Printer	Another type of 3D printer. The extruder moves vertically while the base/print plate remains in its original location.
Extruder	The part of the 3D printer that melts and places the plastic. It is analogous to a hot glue gun.
Filament	The term for the plastic that is used to create the objects.
G-code	The commands sent from a master computer to a CNC machine. The CNC machine then interprets these instructions as motor movements.
Github	A remote git repository website that allows users to store their code at a central location.
GUI	A graphical user interface. Anything that allows users to interact with a program using graphics instead of commandline tools.
HAL file	Hardware abstraction layer files are used to describe how to send instructions to low level devices.
INI file	INI is a file format standard that is often used for configuration files.
LinuxCNC	An operating system designed to run CNC machines. It comes with many useful algorithms such as path planning to make controlling CNC machines easy.
MachineFace	A machine kit user interface designed for mobile and handheld devices.

MachineKit	A fork of LinuxCNC modified to be able to run on more types of hardware including the ARM processor which powers the Beaglebone Black.
MachineTalk	Interface with MachineKit remotely
QT Modeling Language (QML)	A markup language for the QT framework. Many of parts of the standard MachineKit UI and Machineface are written in QML
Raspberry Pi	A cheap single board computer that contains between 256MB and 512MB of RAM. It is capable of running a Linux OS.
ReactJS	An open-source JavaScript library which provides a way of expressing a view without coupling the model or controller.
Reagent	Reagent provides an interface between ClojureScript and ReactJS for JavaScript. It makes using React very pleasurable in ClojureScript.
Single Board Computer (SBC)	A fully functional computer that fits on a single circuit board. It should contain a microprocessor, memory, and IO. It may have Wifi or ethernet capabilities.
MachineFace	An existing user interface for Machinekit
MachineTalk	MachineKit module that allows remote communication while operating in real-time mode.
ProtocolBuffer	Platform-neutral format for serializing structured data. Highly supported languages include C/C++, Python, and Web(JavaScript).

9. References

Practical 3D Printers
This book details everything from how to build a 3D printer to calibration and making models. It teaches the basics of modeling in order to create 3D printed objects. It also includes detailed examples of basic 3D model creation. In addition to being a practical guide, a technical view of the hardware of differing 3D printers is also provided. A major strength of the book is that it does not focus on a particular hardware configuration which allows for a deeper understanding of the process.

LinuxCNC Manual
This reference is the complete documentation of the LinuxCNC project; the project that MachineKit was forked from. Although the document is not specific to MachineKit, MachineKit and LinuxCNC are still similar enough so that this manual still provides a good reference. It is highly useful because of MachineKit's lack of update to date documentation. It contains detailed information on how to use the various GUIs, configuration, and G code programming among many other projects. Congratulations lucky reader! For who knows what reason, you are probably the only person who will ever read this thesis. The author sure didn't. I hope this note brings a smile to your face. Have a nice day. The manual is filled with

technical details, reasons for decisions, information about the organization of the source code, and other topics.

Serialization Framework Comparison
This reference is a comparison of three serialization frameworks: Google's Protocol Buffer, Apache Thrift, and Apache Avro. It explains the differences in handling serialization and deserialization by the different frameworks. Protocol Buffer and Thrift use generators that create files for a variety of programming languages (C++, Java, Python, JavaScript, Haskell, Ruby, Perl, etc). Avro includes the schema of the data in JSON format to avoid generators (trading space in the serialized object to avoid schema versioning). It also compares serialization speeds, deserialization speeds, and serialized object sizes.

Reagent Conference Talk
This reference is an introduction to frontend programming with ClojureScript using Reagent. It provides a good overview of how React, a popular and powerful JavaScript framework, can be used to create web applications written in ClojureScript (code written in Clojure, but compiled to JavaScript). This resource has a basic explanation of how React uses reusable, stateful components and the component lifecycle. The rest of the resource provides code samples and an example web application built using ClojureScript with Reagent (a ClojureScript wrapper for React). Additionally, it discusses the performance of ClojureScript and Reagent.

Reagent Tutorial
This reference is a technical tutorial on creating a simple web application using ClojureScript (code written in Clojure, but compiled to JavaScript) and Reagent (ClojureScript wrapper for the React framework). It does not assume the reader is familiar with ClojureScript, so there is a small tutorial on ClojureScript, and additionally provides external links to more in-depth tutorials. It goes on to introduce Reagent and how it maps ClojureScript to React. It also discusses the role of Atoms in Reagent as a way of handling state. It concludes with a discussion on Ratoms which are Atoms designed for Reagent components that allow the Reagent library to handle component re-rendering as React does.

Appendix – Source Code

project.clj

```
(defproject machineview "0.1.0-SNAPSHOT"
  :description "A clojurescript application for monitoring
machinekit running on a beaglebone."
  :url "http://joshcurtis.github.io/MachineView/"
  :license {:name "Eclipse Public License"
            :url "http://www.eclipse.org/legal/epl-v10.html"}

  :min-lein-version "2.5.0"

  :dependencies [[org.clojure/clojure "1.7.0"]
                 [org.clojure/clojurescript "1.7.228"]
                 [codox "0.9.4"]
                 [binaryage/devtools "0.6.0"]
                 [hiccup "1.0.5"]
```

```
                 [compojure "1.5.0"]
                 [cljs-http "0.1.40"]
                 [javax.servlet/servlet-api "2.5"]
                 [figwheel-sidecar "0.5.2"]
                 [com.cemerick/piggieback "0.2.1"]
                 [cljsjs/react-dom "0.14.3-1"]
                 [cljsjs/react-dom-server "0.14.3-0"]
                 [cljsjs/bootstrap "3.3.6-0"]
                 [cljsjs/filesaverjs "1.1.20151003-0"]
                 [cljsjs/d3 "3.5.16-0"]
                 [cljsjs/three "0.0.72-0"]
                 [reagent "0.6.0-alpha" :exclusions
[cljsjs/react]]]

  :plugins [[lein-figwheel "0.5.2"]
            [lein-doo "0.1.6"]
            [lein-codox "0.9.4"]
            [lein-cljsbuild "1.1.3" :exclusions
[[org.clojure/clojure]]]]

  :source-paths ["src/clj" "src/cljs" "test/clj"]

  :clean-targets ^{:protect false}
["resources/public/js/compiled" "target"]

  :profiles {:clj {:codox {:soure-paths ["src/clj"]
                           :output-path "clj_doc"}}
             :cljs {:codox {:language :clojurescript
                            :source-paths ["src/cljs"]
                            :output-path "cljs_doc"}}}

  :doo {:build "test"}

  :cljsbuild {:builds
              [{:id "dev"
                :source-paths ["src"]

                ;; If no code is to be run, set :figwheel true
for continued automagical reloading
                :figwheel {:on-jsload "app.core/start"}

                :compiler {:main app.core
                           :asset-path "js/compiled/out"
                           :output-to
"resources/public/js/compiled/core.js"
                           :output-dir
"resources/public/js/compiled/out"
```

```
                            :foreign-libs [{:file
"resources/public/js/react-d3.js"
                                           :requires
["cljsjs.d3"

"cljsjs.react"]
                                           :provides
["cljsjs.rd3"]}
                                          {:file
"resources/public/js/TrackballControls.js"
                                           :requires
["cljsjs.three"]
                                           :provides
["cljsjs.trackball-controls"]}]
                            :source-map-timestamp true}}

               {:id "test"
                :source-paths ["src" "test"]

                :compiler {:main app.runner
                           :output-to
"resources/public/js/compiled/test.js"
                           :output-dir
"resources/public/js/compiled/test"
                           :foreign-libs [{:file
"resources/public/js/react-d3.js"
                                           :requires
["cljsjs.d3"]
                                           :provides
["cljsjs.rd3"]}
                                          {:file
"resources/public/js/TrackballControls.js"
                                           :requires
["cljsjs.three"]
                                           :provides
["cljsjs.trackball-controls"]}]}}


               ;; This next build is an compressed minified
build for
               ;; production. You can build this with:
               ;; lein cljsbuild once min
               {:id "min"
                :source-paths ["src"]
                :compiler {:output-to
"resources/public/js/compiled/core.js"
```

```
                            :foreign-libs [{:file
"resources/public/js/react-d3.js"
                                           :requires
["cljsjs.d3"]
                                           :provides
["cljsjs.rd3"]}
                                          {:file
"resources/public/js/TrackballControls.js"
                                           :requires
["cljsjs.three"]
                                           :provides
["cljsjs.trackball-controls"]}]
                            :externs
["resources/public/js/externs.js"]
                            :main app.core
                            :optimizations :advanced
                            :pretty-print false}}]}

  :figwheel {;; :http-server-root "public" ;; default and
assumes "resources"
             :server-port 3000 ;; default
             ;; :server-ip "127.0.0.1"

             :css-dirs ["resources/public/css"] ;; watch and
update CSS

             ;; Start an nREPL server into the running figwheel
process
             ;; :nrepl-port 7888

             ;; Server Ring Handler (optional)
             ;; if you want to embed a ring handler into the
figwheel http-kit
             ;; server, this is for simple ring servers, if this
             ;; doesn't work for you just run your own server :)
             ;; :ring-handler hello_world.server/handler
             :ring-handler server.core/handler

             ;; To be able to open files in your editor from the
heads up display
             ;; you will need to put a script on your path.
             ;; that script will have to take a file path and a
line number
             ;; ie. in  ~/bin/myfile-opener
             ;; #! /bin/sh
             ;; emacsclient -n +$2 $1
             ;;
```

```
              ;; :open-file-command "myfile-opener"

              ;; if you want to disable the REPL
              ;; :repl false

              ;; to configure a different figwheel logfile path
              ;; :server-logfile "tmp/logs/figwheel-logfile.log"
              }

  :repl-options  {:nrepl-middleware [cemerick.piggieback/wrap-
cljs-repl]
                  :init (do
                          (use 'figwheel-sidecar.repl-api)
                          (start-figwheel!)
                          (println "Run (cljs-repl) to connect
to the cljs repl"))})
```

bb_server.py

```
#!/usr/bin/env python2
from flask import Flask, send_from_directory
import edn_format as edn

from glob import glob
import datetime, os, subprocess, zmq, time, random

from machinetalk.protobuf.message_pb2 import Container
from machinetalk.protobuf.types_pb2 import MT_PING


app = Flask(__name__)
app.debug = True

login_username = 'machinekit'
login_password = 'kit'

# stolen code for crossdomain
from datetime import timedelta
from flask import make_response, request, current_app
from functools import update_wrapper

def crossdomain(origin=None, methods=None, headers=None,
                max_age=21600, attach_to_all=True,
                automatic_options=True):
    if methods is not None:
        methods = ', '.join(sorted(x.upper() for x in methods))
```

```
    if headers is not None and not isinstance(headers,
basestring):
        headers = ', '.join(x.upper() for x in headers)
    if not isinstance(origin, basestring):
        origin = ', '.join(origin)
    if isinstance(max_age, timedelta):
        max_age = max_age.total_seconds()

    def get_methods():
        if methods is not None:
            return methods

        options_resp =
current_app.make_default_options_response()
        return options_resp.headers['allow']

    def decorator(f):
        def wrapped_function(*args, **kwargs):
            if automatic_options and request.method ==
'OPTIONS':
                resp =
current_app.make_default_options_response()
            else:
                resp = make_response(f(*args, **kwargs))
            if not attach_to_all and request.method !=
'OPTIONS':
                return resp

            h = resp.headers

            h['Access-Control-Allow-Origin'] = origin
            h['Access-Control-Allow-Methods'] = get_methods()
            h['Access-Control-Max-Age'] = str(max_age)
            if headers is not None:
                h['Access-Control-Allow-Headers'] = headers
            return resp

        f.provide_automatic_options = False
        return update_wrapper(wrapped_function, f)
    return decorator

status = {"ok?": True,
          "mk_running": False,
          "resolving_services": False}

config_root = os.path.expanduser("~/machinekit/configs")
service_log_path = os.path.expanduser("~/Desktop/services.log")
```

```
service_log = open(service_log_path, 'a')
mklauncher = None
configserver = None
linuxcnc = None
resolver = None

def clear_services_log():
    global service_log_path
    open(service_log_path, 'w').write('cleared\n')
clear_services_log()

@app.route("/status", methods=['GET'])
@crossdomain(origin="*")
def route_status():
    global status
    return edn.dumps(status)

@app.route("/login", methods=['POST', 'OPTIONS'])
@crossdomain(origin="*")
def route_login():
    global status
    if request.method == 'OPTIONS':
        return edn.dumps({})
    if request.method == 'POST':
        if request.form.get('password') == login_password:
            return '{:authenticated true :username ' +
login_username + '}'
        else:
            return '{:authenticated false}'

@app.route("/configs", methods=['GET'])
@crossdomain(origin="*")
def route_configs():
    global config_root
    config_dirs = glob("{}/*/".format(config_root))
    files = map(lambda d: glob("{}*".format(d)), config_dirs)
    get_f = lambda s: s.split('/')[-1]
    m_keys = map(lambda p: p.split('/')[-2]+'/', config_dirs)
    m_vals = map(lambda fs: map(get_f, fs), files)
    m = dict(zip(m_keys, m_vals))
    return edn.dumps(m)

@app.route("/config", methods=['GET', 'PUT', 'DELETE',
'OPTIONS'])
@crossdomain(origin="*")
def route_configs_file():
    global config_root
```

```
    path = "{}/{}".format(config_root, request.args.get('path'))
    if request.method == 'GET':
        txt = open(path).read()
        return edn.dumps({"contents": txt})
    elif request.method == 'PUT':
        fp = open(path, 'w')
        fp.write(request.form.get('contents'))
        return edn.dumps({})
    elif request.method == 'OPTIONS': # http/delete calls
options then delete
        return edn.dumps({})
    elif request.method == 'DELETE':
        os.remove(path)
        return edn.dumps({})

@app.route("/resolve", methods=['GET'])
@crossdomain(origin="*")
def route_resolve():
    global service_log
    global resolver
    cmd = 'python /home/machinekit/Desktop/resolve.py'.split()
    stop_process(resolver)
    resolver = subprocess.Popen(cmd, stdout=service_log,
stderr=service_log)
    if not status['resolving_services']:
        status['resolving_services'] = True
    return edn.dumps(status)

@app.route("/services_log", methods=['GET'])
@crossdomain(origin="*")
def route_services_log():
    global service_log_path
    text = open(service_log_path).read()
    return edn.dumps({"log": text})

@app.route("/run_mk", methods=['GET'])
@crossdomain(origin="*")
def route_run_mk():
    global status
    run_mk()
    if not status["mk_running"]:
        status["mk_running"] = True
    return edn.dumps(status)

def run_mk():
    global mklauncher
    global configserver
```

```
    global linuxcnc
    cmd1 = 'configserver /home/machinekit/Desktop -d'.split()
    configserver = subprocess.Popen(cmd1)
    return edn.dumps(status)
    # cmd2 = 'linuxcnc
/home/machinekit/machinekit/configs/ARM.BeagleBone.CRAMPS/CRAMPS
_REMOTE.ini -d'.split()
    # linuxcnc = subprocess.Popen(cmd2)

@app.route("/stop_mk", methods=['GET'])
@crossdomain(origin="*")
def route_stop_mk():
    global status
    global mklauncher
    global configserver
    global linuxcnc
    stop_process(mklauncher)
    stop_process(configserver)
    stop_process(linuxcnc)
    stop_process(resolver)
    if status["mk_running"]:
        status["mk_running"] = False
    return edn.dumps(status)

def stop_process(process):
    if process != None:
        if process.returncode == None:
            process.terminate()

@app.route("/ping/<port>", methods=['GET'])
@crossdomain(origin="*")
def ping(port):
    send_data(port, MT_PING)
    return edn.dumps(status)

def send_data(port, msg_type):
    msg = Container()
    msg.type = msg_type
    context = zmq.Context()
    dealer = context.socket(zmq.DEALER)
    dealer.identity = 'batman'
    hostname = 'tcp://localhost:' + str(port)

    dealer.connect(hostname)
    dealer.send(msg.SerializeToString())
    return
```

```
@app.route("/running", methods=['GET'])
@crossdomain(origin="*")
def running():
    status['mk_running'] = (configserver != None and
configserver.poll() == None)
    return edn.dumps(status['mk_running'])

# TODO - "t", gives a large value, currently overwritten in
client to improve
# usefulness
sim_pos = 0.0
@app.route("/measure", methods=['GET'])
@crossdomain(origin="*")
def route_measure():
    global sim_pos
    t = time.time() / 1000.0
    sim_pos += 0.1
    if sim_pos > 1.0:
        sim_pos = 0.0
    return edn.dumps({"t": t,
                      "Ext-0": random.random(),
                      "Ext-1": random.random() + 1.0,
                      "Ext-2": random.random() + 2.0,
                      "Axis-0-x": sim_pos,
                      "Axis-0-y": 0.0,
                      "Axis-0-z": 0.0,
                      "Axis-0-a": 0.0,
                      "Axis-1-x": -sim_pos,
                      "Axis-1-y": 0.0,
                      "Axis-1-z": 0.0,
                      "Axis-1-a": 0.0,
                      "Axis-2-x": 0.0,
                      "Axis-2-y": sim_pos,
                      "Axis-2-z": 0.0,
                      "Axis-2-a": 0.0,
                      "Axis-3-x": 0.2*(random.random() - 0.5),
                      "Axis-3-y": 0.2*(random.random() - 0.5),
                      "Axis-3-z": max(0.0, 0.2*(random.random()
- 0.5)),
                      "Axis-3-a": 0.0})

@app.route("/js/<path:path>")
def route_js(path):
    return send_from_directory("./../resources/public/js", path)

@app.route("/css/<path:path>")
def route_css(path):
```

```
    return send_from_directory("./../resources/public/css",
path)

@app.route ("/icons/<path:path>")
def route_icons(path):
    return send_from_directory("./../resources/public/icons",
path)

@app.route("/favicon.ico")
def route_favicon():
    return send_from_directory("./../resources/public",
"favicon.ico")

@app.route("/")
def route_index():
    return send_from_directory("./../resources/public",
"index.html")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=3001)
```

app/core.cljs

```
(ns app.core
  "Mounts the application onto the `div` with the id `\"app\"`."
  (:require
   [model.core :as model]
   [controller.core]
   [view.core]
   [cljsjs.bootstrap]
   [app.devtools-setup]
   [view.topbar]
   [widgets.core :as widgets]
   [utils.core :as utils]
   [viz.core :as viz]
   [reagent.core :as r :refer [atom]]))

(defn app
  "Reagent component which describes the app. It is a tab bar
followed by the
  contents of that tab."
  [props]
  (let [tab @(r/cursor model/state [:tab])
        tab-labels @(r/cursor model/state [:tab-labels])]
    [:div.app {}
     [view.topbar/render-topbar {}]
     [widgets/tabs {:labels tab-labels
```

```
                    :id-prefix "tab-navigation-"
                    :selected tab
                    :on-change #(controller.core/set-tab! %1)}]
     [view.core/app-view]]))

(defn ^:export start
  "Renders the application onto the DOM element \"app\""
  []
  (r/render-component [app {}]
                      (.getElementById js/document "app")))

(start)
```

Bbserver/core.cljs

```
(ns bbserver.core
  "Functions to interact with the hardware. Should be removed as
HTTP interface
  isn't complex and cljs-http allows for a synchronous looking
workflow."
  (:require-macros
   [cljs.core.async.macros :refer [go]])
  (:require
   [clojure.string :as string]
   [cljs.reader :as reader]
   [cljs-http.client :as http]
   [cljs.core.async :refer [<!]]))

(defn- bb-wrapper
  ([http-type address callback options]
   (go (let [options (merge {:with-credentials? false :timeout
2000} options)
             resp (<! (http-type address
                                 options))
             {:keys [status error-code body]} resp
             body (reader/read-string body)]
         (callback status error-code body))))
  ([http-type address callback] (bb-wrapper http-type address
callback {})))

(defn- bb-get
  ([address callback]
  (bb-get address callback {}))
  ([address callback options]
  (bb-wrapper http/get address callback options)))
```

```
(defn- bb-put
  [address callback options]
  (bb-wrapper http/put address callback options))

(defn- bb-post
  [address callback options]
  (bb-wrapper http/post address callback options))

(defn- bb-delete
  ([address callback]
  (bb-delete address callback {}))
  ([address callback options]
  (bb-wrapper http/delete address callback options)))

(defn- build-address
  [hostname port route]
  (str "http://" hostname \: port route))

(defn status
  [hostname callback]
  (bb-get (build-address hostname 3001 "/status") callback))

(defn login
  [hostname password callback]
  (bb-post (build-address hostname 3001 "/login") callback
    {:form-params {:password password}}))

(defn configs
  [hostname callback]
  (bb-get (build-address hostname 3001 "/configs") callback))

(defn get-file
  [hostname config filename callback]
  {:pre [(string/includes? config \/)]}
  (bb-get (build-address hostname 3001 "/config") callback
    {:query-params {:path (str config filename)}}))

(defn put-file
  [hostname config filename contents callback]
  {:pre [(string/includes? config \/) (string? contents)]}
  (bb-put (build-address hostname 3001 "/config") callback
    {:query-params {:path (str config filename)}
     :form-params {:contents contents}}))

(defn delete-file
  [hostname config filename callback]
```

```
  {:pre [(string/includes? config \/)]}
  (bb-delete (build-address hostname 3001 "/config") callback
    {:query-params {:path (str config filename)}}))

(defn get-services-log
  [hostname callback]
  (bb-get (build-address hostname 3001 "/services_log")
callback))

(defn resolve-services
  [hostname callback]
  (bb-get (build-address hostname 3001 "/resolve") callback))

(defn run_mk
  [hostname callback]
  (bb-get (build-address hostname 3001 "/run_mk") callback))

(defn stop_mk
  [hostname callback]
  (bb-get (build-address hostname 3001 "/stop_mk") callback))

(defn ping
  [hostname port callback]
  (bb-get (build-address hostname 3001 (str "/ping/" port))
callback))

(defn running
  [hostname callback]
  (bb-get (build-address hostname 3001 "/running") callback))
```

Controler/remote_monitor.cljs

```
(ns controller.remote-manager
  "Actions for connecting, disconnecting, and getting the state
of the
  hardware. The connecting and disconnecting should be moved."
  (:require
   [model.core :as model]
   [controller.ini-editor]
   [controller.text-editor]
   [bbserver.core :as bbserver]
   [utils.core :as utils]
   [clojure.string :as string]))

(defonce mt (.-protobuf js/machinetalk))
```

```
(defonce container-types (.-ContainerType (.-message mt)))
(defonce MT_PING (.-MT_PING container-types))
(defonce MT_SHUTDOWN (.-MT_SHUTDOWN container-types))

; Forward declare
(declare
  start-update-configs-and-services-interval
  start-update-running-interval)

(defn set-hostname!
  [name]
  (swap! model/state assoc-in [:connection :hostname] name))

(defn set-password!
  [password]
  (swap! model/state assoc-in [:connection :password] password))

(defn- merge-with-state-connection
  [merge-map]
  (swap! model/state update :connection #(merge %1 merge-map)))

(defn- update-configs-callback
  "Callback for a file list in the machinekit config directory."
  [status error-code body]
  (if (and (= status 200) (some? body))
    (let [dirs (mapv identity (keys body))]
      (swap! model/state assoc :configs {:dirs dirs :contents
body}))))

(defn- update-configs!
  "Requests a file list in the machinekit config directory."
  []
  (let [hostname (get-in @model/state [:connection :hostname])]
    (bbserver/configs hostname update-configs-callback)))

(defn- update-mk-services-callback
  "Callback for MachinkeKit services list."
  [status error-code body]
  (if (and (= status 200) (some? body))
    (swap! model/state assoc :services (utils/parse-service-log
(get body "log")))))

(defn- update-mk-services!
  "Request the ~/Desktop/services.log file which will
  be parsed to update and inform the user what machinekit
  services are available"
  []
```

```
  (let [hostname (get-in @model/state [:connection :hostname])]
    (bbserver/get-services-log hostname update-mk-services-
callback)))

(defn- log-body
  [status error-code body]
  (if (and (= status 200) (some? body))
    (utils/log body)
    (utils/log [:status status :error-code error-code])))

(defn- try-to-launch-resolver!
  "Request that services be resolved."
  []
  (let [hostname (get-in @model/state [:connection :hostname])]
    (do
      (utils/log "Launching Resolver")
      (bbserver/resolve-services hostname log-body))))

(defn- update-running-callback
  [status error-code body]
  (cond
    (or (= error-code :http-error) (= error-code :timeout))
      (merge-with-state-connection {:connected? false
                                    :connection-pending? false
                                    :username nil
                                    :error "Disconnected from
BeagleBone"})
    :else
      (swap! model/state assoc :running? body)
  ))

(defn- update-running!
  []
  "Request to see if MachinkeKit is running."
  (let [hostname (get-in @model/state [:connection :hostname])]
    (bbserver/running hostname update-running-callback)))

(defn- connect-callback
  "Callback for connection/login attempt with the BeagleBone."
  [status error-code body]
  (cond
    (or (= error-code :http-error) (= error-code :timeout))
      (merge-with-state-connection {:connected? false
                                    :connection-pending? false
                                    :username nil
                                    :error "Unable to connect
with host"})
```

```
    (nil? body)
      (merge-with-state-connection {:connected? false
                                    :connection-pending? false
                                    :username nil
                                    :error "Error with response
body"})
    :else
    (let [{:keys [authenticated username]} body]
      (if (not authenticated)
        (merge-with-state-connection {:connected? false
                                      :connection-pending? false
                                      :username nil
                                      :error "Incorrect
password"})
        (do
          (merge-with-state-connection {:connected? true
                                        :connection-pending?
false
                                        :username username
                                        :error nil})
          (update-configs!)
          (try-to-launch-resolver!)
          (start-update-configs-and-services-interval)
          (start-update-running-interval))))
  ))

(defn connect!
  "Request authorization to access the BeagleBone."
  []
  (let [{:keys [connection]} @model/state
        {:keys [hostname username password]} connection]
    (merge-with-state-connection {:connected? false
                                  :connection-pending? true
                                  :username nil
                                  :error nil})
    (bbserver/login hostname password connect-callback)))

(defn disconnect!
  "Disconnect from the BeagleBone and return to the login
screen."
  []
  (merge-with-state-connection {:connected? false
                                :connection-pending? false
                                :username nil
                                :error nil})
  (utils/clear-interval "update-configs-and-services")
  (utils/clear-interval "update-running"))
```

```
(defn run-mk!
  []
  (if (-> @model/state :connection :connected?)
    (let [hostname (get-in @model/state [:connection
:hostname])]
      (do
        (bbserver/run_mk hostname log-body)
        (swap! model/state assoc :running? true)))))

(defn shutdown-mk!
  []
  (if (-> @model/state :connection :connected?)
    (let [hostname (get-in @model/state [:connection
:hostname])]
      (do
        (bbserver/stop_mk hostname log-body)
        (swap! model/state assoc :running? false)))))

(defn- edit-ini!
  [s id]
  (assert (string? s))
  (assert (some? id))
  (controller.ini-editor/load-str! id s)
  (utils/click-element "tab-navigation-INI"))

(defn- edit-unsupported
  [s id]
  (controller.text-editor/load-text! id s)
  (utils/click-element "tab-navigation-Text"))

(def edit-callbacks {"ini" edit-ini!})

(defn edit-file!
  [config filename]
  (let [extension (utils/file-ext filename)
        hostname (get-in @model/state [:connection :hostname])
        callback (get edit-callbacks extension edit-unsupported)
        callback #(callback (get %3 "contents") [:remote
filename])]
    (bbserver/get-file hostname config filename callback)))

(defn upload-file!
  [config filename contents]
  (let [hostname (get-in @model/state [:connection :hostname])]
    (bbserver/put-file hostname config filename contents update-
configs!)))
```

```
(defn download-file!
  [config filename]
  (let [hostname (get-in @model/state [:connection :hostname])
        callback #(utils/save-file (get %3 "contents")
filename)]
    (bbserver/get-file hostname config filename callback)))

(defn delete-file!
  [config filename]
  (let [hostname (get-in @model/state [:connection :hostname])]
    (bbserver/delete-file hostname config filename update-
configs!)))

(defn ping
  []
  (let [hostname (get-in @model/state [:connection :hostname])
        port (get-in @model/state [:services :config])]
    (bbserver/ping hostname port #(utils/log "Pinging
MachineKit"))))

; Create an update interval while the user is connected that
; updates configs and mk-services
(defn- start-update-configs-and-services-interval
  []
  (utils/set-interval "update-configs-and-services"
    #(let [{:keys [connection]} @model/state]
      (if (:connected? connection)
        (do
          (update-configs!)
          (update-mk-services!))))
    500))

; Create an update interval while the user is connected that
; updates whether MachineKit is running
(defn- start-update-running-interval
  []
  (utils/set-interval "update-running"
    #(let [{:keys [connection]} @model/state]
      (if (:connected? connection)
        (update-running!)))
    500))

; Create a debug logging interval
(utils/set-interval "debug-state"
  #(do
    (utils/log (str "Services: " (:services @model/state)))
```

```
   )
  5000)
```

Utils/core.cljs

```
(ns utils.core
  "Generic utility functions."
  (:require
   [cljsjs.filesaverjs]
   [clojure.set]
   [clojure.string :as string]))

(defn map-do
  "Not lazy version of map."
  [f coll]
  (doall (map f coll)))

(defn timestamp-to-str
  [t]
  (-> t
      js/Date.
      .toString))

(defn alert
  "Shows an alert box with the provided arguments displayed as strings."
  ([arg] (js/alert (str arg)))
  ([& args] (js/alert (str args))))

(defn time-seconds
  "Returns a timestamp, in seconds."
  []
  (-> js/Date .now (/ 1000.0)))

(defn append-line
  "Appends new-line to old-lines. If old-lines is blank or nil, then new-line
is
  returned"
  [old-lines new-line]
  (if (or (nil? old-lines) (string/blank? old-lines))
    new-line
    (str old-lines \newline new-line)))

(defn click-element
  "The DOM element with the id `id` is clicked."
  [id]
  (.click (.getElementById js/document id)))

(defn element-value
  "The value of the DOM element with the `id`."
  [id]
  (.-value (.getElementById js/document id)))

(defn read-file
  "`js-file-obj` must be a JavaScript `File` object and callback must be a
  function that takes a `string`. The contents of the `js-file-obj` as a
```

```
  `string` are passed to the provided `callback`. If it fails, then nothing
  happens."
  [js-file-obj callback]
  (let [reader (js/FileReader.)
        cback #(callback (-> %1 .-target .-result))]
    (aset reader "onload" cback)
    (.readAsText reader js-file-obj)))

(defn remove-idx
  "Given a `vector v`, `v` is returned with item at index `idx` removed."
  [v idx]
  (assert (vector? v) "v is not a vector")
  (vec (concat (subvec v 0 idx) (subvec v (inc idx)))))

(defn save-file
  "Saves the contents of `text` with a target filename of `filename`."
  [text filename]
  (assert (string? text))
  (assert (string? filename))
  (let [blob (js/Blob. #js [text] #js {"type" "text/plain;charset=utf-8"})]
    (js/saveAs blob filename)))

(defn toggle-membership
  "Given a `set s`, element `v` is removed if it is in the set, or added if
it
  is not."
  [s v]
  (assert (set? s))
  (if (contains? s v)
    (clojure.set/difference s #{v})
    (clojure.set/union s #{v})))

;; file string helpers

(defn dir?
  "Returns true if the string represents a directory. This is checked by
looking
  for the / character at the end."
  [s]
  (assert (string? s))
  (string/ends-with? s \/))

(defn fname-from-path
  "Returns the filename from the path, this is the element after the last /."
  [path]
  (last (string/split path \/)))

(defn file-ext
  "Returns the extension of the file. If there is no extension, then the
blank
  string is returned. The extension of a file is whatever substring comes
after
  the .(dot)."
  [s]
  (assert (string? s))
  (if (and (string/includes? s \.) (not (dir? s)))
    (-> s (string/split \.) last string/lower-case)
```

```
    ""))

;; id stuff

(defonce unique-int-counter (atom 0))

(defn unique-int
  "Returns a unique integer at every call. The current implementation starts
at
  0 and counts upward."
  []
  (swap! unique-int-counter inc))

(defn unique-dom-id
  "Returns a unique string that can be used to label dom elements"
  [prefix]
  (str prefix (unique-int)))

(defn gen-unique-id
  "Generates an id so that it does not exist in ids. An id is a vector tuple
of
  structure [any string]. To make the id unique, the string is prefixed with
  `copy-` until it is not found in the set `ids`."
  [ids id]
  (loop [id id]
    (if (contains? ids id)
      (recur (update id 1 #(str "copy-" %1)))
      id)))

;; intervals that are reload friendly

(defonce intervals (atom {}))

(defn active-intervals
  "Returns a list of the ids of the currently running intervals that were
  created by the `set-interval` function."
  []
  (keys @intervals))

(defn clear-interval
  "Clears the interval with the specified id. If no such id exists, then
nothing
  happens."
  [id]
  (assert (string? id))
  (let [i (get @intervals id)]
    (if (some? i) (do
                    (js/clearInterval i)
                    (swap! intervals dissoc id)))))

(defn clear-all-intervals
  "Clears all intervals."
  []
  (map clear-interval (active-intervals)))

(defn set-interval
  "Calls function `f` in an interval of approximately `milliseconds`
```

```
  milliseconds. If an interval with the given `id` already exists, then that
  interval is cleared and a new one is created."
  [id f milliseconds]
  (assert (string? id))
  (assert (fn? f))
  (assert (some? milliseconds))
  (clear-interval id)
  (swap! intervals assoc id (js/setInterval f milliseconds)))

(defn parse-removed-line
  [line]
  (let [tokens (map (comp keyword string/lower-case) (string/split line " "))
        launchercmd (nth tokens 3)
        launcher (nth tokens 4)]
    (cond
      (= :launcher launcher) launcher
      (= :launchercmd launchercmd) launchercmd
      :else :error)))

(defn parse-resolved-line
  [line]
  (let [[_ service address] (string/split line " ")
        [_ _ port] (string/split address ":")]
    {(keyword service) port}))

(defn parse-service-lines
  "Takes a list of lines declaring or removing a service.
  Outputs a dictionary of current services and their ports.
  Return an empty dictionary if no services are available."
  [[line & lines] services]
  (if (some? line)
    (let [update (first (string/split line " "))]
      (cond
        (= update "removed")  (parse-service-lines lines (dissoc services
(parse-removed-line line)))
        (= update "resolved") (parse-service-lines lines (merge services
(parse-resolved-line line)))))
    services))

(defn clean-service-log
  "Removes trailing and leading whitespace from a log
  Splits the log into lines
  Keeps only lines that are service updates"
  [log]
  (let [lines (->> log string/split-lines (map string/trim))]
    (filter #(or (string/starts-with? %1 "resolved")
                 (string/starts-with? %1 "removed"))
            lines)))

(defn parse-service-log
  "Takes a log of output from the resolve.py script
  and returns a dictionary of available services with their ports"
  [log]
  (parse-service-lines (clean-service-log log) {}))

(defn log
  "Alias for the javascript function console.log
```

```
  Stringifies object before printing them for convenience"
  ([x] (.log js/console (str x)))
  ([& args] (.log js/console (str args))))

(defn encode-buffer
  "TODO: Handle more data"
  [type]
  (let [mt (.-protobuf js/machinetalk)
        container (.-Container (.-message mt))
        container-types (.-ContainerType (.-message mt))
        encoded (.encode container type)
        limit (.-limit encoded)
        buffer (.-view encoded)
        sliced (map #(aget buffer %) (range 3))]
    sliced))
```