NORTHWEST NAZARENE UNIVERSITY

Interacting With a User Interface Using Robotics: a C# Library and Python Based Server

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF ARTS

Timothy Scott Mong
2017

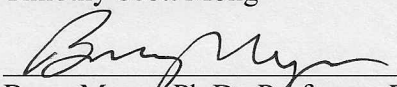
THESIS

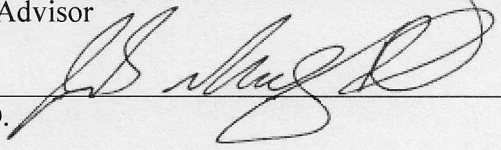
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF ARTS

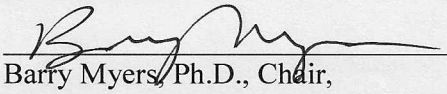
Timothy Scott Mong
2017

Interacting With a User Interface Using Robotics: a C# Library and Python Based Server

Author: 
Timothy Scott Mong

Approved: 
Barry Myers, Ph.D., Professor, Department of Mathematics and Computer Science, Faculty Advisor

Approved: 
Jacob Mong, D.O.
Second Reader

Approved: 
Barry Myers, Ph.D., Chair,
Department of Mathematics & Computer Science

Abstract

Interacting With a User Interface Using Robotics: a C# Library and Python Based Server
MONG, TIMOTHY (Department of Mathematics and Computer Science), MYERS, DR. BARRY (Department of Mathematics and Computer Science).

In the Reliable Deployment Lab at HP (RDL), automation is a vital aspect of testing. Currently, web requests perform all the automation, but there is no mechanism for automated physical interaction. Since humans do not typically interact with a device via web-requests, RDL assigned the developer a prototype project that manipulates a device screen physically. At the beginning of the project, the robotics package could move and press a stylus in a coordinate system native to the robot. The developer created a C# library that sends Move and Push requests to the robot via HTTP. The library also converts user interface coordinates into the native robot coordinates. To test the library and provide demonstration capabilities, the developer also created a test harness and WinForms demo utility.

The robot hosts an HTTP server and listens for HTTP POST requests that solicit interactions in a network environment. The developer also created the Modular Architecture Robot Server (MARS) for selecting and mounting additional python modules on the robot for exposure as a resource. The robot can perform standard workflows but requires human aid for coordinate discovery. The project was a success but likely will not be implemented as a testing standard.

Acknowledgments

I would like to thank my wife and son for supporting me throughout my pursuit of education. I would also like to thank HPI for allowing me to use my intern project in my Senior Project. Finally, I would like to thank my professors who have instructed me and graded so so many of my homework assignments.

Table of Contents

Figures and Equations

Introduction

The student developed a C# client, Coordinate Conversion Class, and Python Server to facilitate automated physical testing of devices. The project is multifaceted involving several independent but related applications. In the following subsections, the student discusses the background, target audience, and project requirements.

Product Overview

The product consists of a C# library with a demo application and test harness. Additionally, there is a Python Server for communication with the robot. Together, the library and robot server can perform predetermined workflows on a device screen.

Background

In the Reliable Deployment Lab at HP (RDL), automation is a vital aspect of testing. Currently, web requests perform nearly all the automation; therefore, physical interaction does not. Because humans do not typically interact with a device via web-requests, RDL assigned the developer a prototype project that manipulates a device screen physically.

Although there is a vast and complex infrastructure in place for reserving devices, the tests interact with a device via web requests initiated by a library call. To maintain uniformity of design, the physical automation of a device should launch by a call to a library.

At the onset of the project, RDL provided the student a basic robot that had the functionality to press at a given coordinate pair. What the robot lacked was a client to direct and communicate the move/press requests. Furthermore, the coordinates for the robot are not compatible with a standard device screen and no conversion tool was provided.

Target Audience

The direct target audience is the Reliable Deployment Lab itself. All current customers

that rely on quality control from RDL form an indirect audience. The goal is to provide another tool to testers and test developers in order to add another dimension of quality to a test. Since there is currently automation available to testers and developers, the robotics project is designed to fill a niche need for bugs that only manifest during manual workflows.

Project Requirements

There is one overarching project need: A client must be able to press the device screen through the robot. The following is an explicit list of implicit requirements:

1. A client must communicate with the robot
2. The coordinates of the device must match the coordinates of the robot
3. The client must be in a language compatible with the tests
4. Since the robot runs on a Linux operating system, the server must be Linux compatible
5. The robot must perform at a reasonable speed, approximating that of a human

All of these requirements are necessary for the project to be useful to the target audience. RDL provided the student leeway to implement the applications as long as the student met the general requirements.

Development

As a methodology, the student started with an analysis of the problem and then the implementation of the chosen solution. The following subsections include the requirements-gathering, evaluation of technologies, and how the project would interact with the larger system.

Analysis

The student performed an extensive analysis to discover and identify the problem and an appropriate strategy to find a solution. In other words, the student reviewed numerous sources to identify both the functional and non-functional project requirements. There are various considerations and potential solutions; however, the student finally chose a C# library and a python based server as they provided the best balance of available time and platform compatibility.

Requirements Gathering. The requirement gathering phase consisted of interviews with the supervisor of the student, senior developers, and the robotics lab. The meeting with the supervisor determined the overall goals of the project. The interviews with senior developers revealed the context and working environment for any developed applications. The interviews with the robotics lab helped discover the pre-existing capabilities of the robot. Furthermore, the student reviewed example workflows and test scenarios to find the most useful implementation of the library for the given environment.

System Interaction. The robotics platform and client interact within a scalable environment. Additionally, the overall system includes the capability to reserve a device and query device information for initialization in the test. Any real integration, therefore, required storing robot information in the same database as the device information. For example, if a test needs a printer to run, it would be useful to know if that printer has an association with a robot.

Therefore, a printer may have one or zero robots associated with itself. As an administrator or tester moves a robot from printer to printer, the mover is responsible for appropriately updating the database just as would happen if something changed for a printer.

Previous Languages. Since the automation that was already in place used library calls within a test, the student decided that this method would be an exceptional way to introduce added functionality. Therefore, the student developed a prototype C# library that provides calls to physical presses instead of virtual presses.

Framework and Language Selection

Since the current tests and framework are in C#, the student decided to implement the client in C# as well. However, the student developed the server in Python with CherryPy as a server framework. The following sections discuss the decisions in more depth.

Python. The robot operates on Lubuntu, a version of Linux. Because of the operating system, the student considered either Python or Java. Because the student had significant previous experience in Python and because of the syntactic power of Python, this was the natural choice for the project. Python, like Java, is platform independent and therefore a compatible choice with the given operating system.

CherryPy. According to CherryPy.org, Cherrypy is a Hypertext Transfer Protocol (HTTP) server framework that provides basic hosting functionality in an intuitive and well-documented manner (2016). Because of the ease of use, as well as the extensive documentation, the student chose CherryPy as the server framework. The student also considered communicating directly through sockets, but found CherryPy to be more suitable for rapid development.

Microsoft C#. The student chose Microsoft C# because the existing frameworks use C# and because of the vast support community that exists for C# development. Additionally, the

Graphical User Interface support for C# made the development of demo applications much more intuitive. While any Managed Code language would technically be compatible, the student found that working in the same language as the existing framework eased compatibility and maintained a consistent paradigm.

Development of Modular Architecture Robot Server

The Modular Architecture Robot Server (MARS) is a server that can host multiple modules that a user wishes to expose as part of a Hypertext Transfer Protocol (HTTP) based Application Programming Interface(API). Therefore, the server has the capability of module selection and discovering contents of modules for exposure to the network via HTTP.

Modules. The CherryPy framework mounts functionality as a class with the following HTTP methods:

- GET
- POST
- PUT
- DELETE

Each class within a module conforms to a general format. Figure 1 illustrates the format.

```
import cherrypy
import yourextramodules #(an api that your GET can call into, for example)
class myFunction:
        exposed = True
        def GET(your parameters here):
                #your code here
        def POST:
                #your code here
        def PUT:
                #your code here
        def DELETE:
                #your code here
```

Figure 1. Module Format

A module may have multiple classes if each and every class follows the given format. In this way, CherryPy may mount an instance of each class.

Dynamic Import. What happens if modules are not hard-coded in the server application itself? Or, what if the user may select modules at run time? Using the Python Imp library, the student wrote a function that inspects selected modules and discovers classes to mount dynamically. Because of the dynamic nature, the student does not need to modify server code when more functionality is required, just the affected module. Figure 2 illustrates the discovery and mounting of classes within a dynamically imported Python module.

```
#Method: mountModules
#Functionality: This method looks inside the selected modules to expose the specific functionality of each on cherrypy
#Parameters: self (Python does this automatically)
def mountModules(self):
    for name in self._moduleList:

        moduleName = ""
        #for each module, get the classes therein
        for property, description in getmembers(name):
            if property == "__name__":
                moduleName = description
        print moduleName
        classes = getmembers(name, isclass)

        #mount an instance of each class
        for aclass in classes:

            #strip off the consistent extra characters that inspection returns
            #and make sure that there are not identical class names
            #Format: ('classname', <class classname.classname at 0xffffffff>)
            justName = str(aclass).split(",")
            justName = justName[0].strip("('")
            URL = self.checkDuplicateClass(justName, moduleName)
            cherrypy.tree.mount(getattr(name, justName)(), '/api/' + URL,
                                {'/':
                                {'request.dispatch': cherrypy.dispatch.MethodDispatcher()}  #This dispatcher expects REST
                                })

    self.logExposedApi()
    #update your changes in the server.conf file to apply custom hosting options
    #cherrypy.config.update(self.conf)    #security configs
    CONFIG = cherrypy.config.update('server.conf')  #user defined configs
    cherrypy.engine.start()
    self.started = True
    cherrypy.engine.block()
```

Figure 2. Mount Dynamic Modules

The student also developed a Graphical User Interface (GUI) for the server using the Tkinter Python library. The GUI is composed of standard components such as buttons, text boxes, and list boxes. The GUI allows a user to select multiple modules and see what it is they are selecting. Figure 3 depicts the GUI with several modules selected.

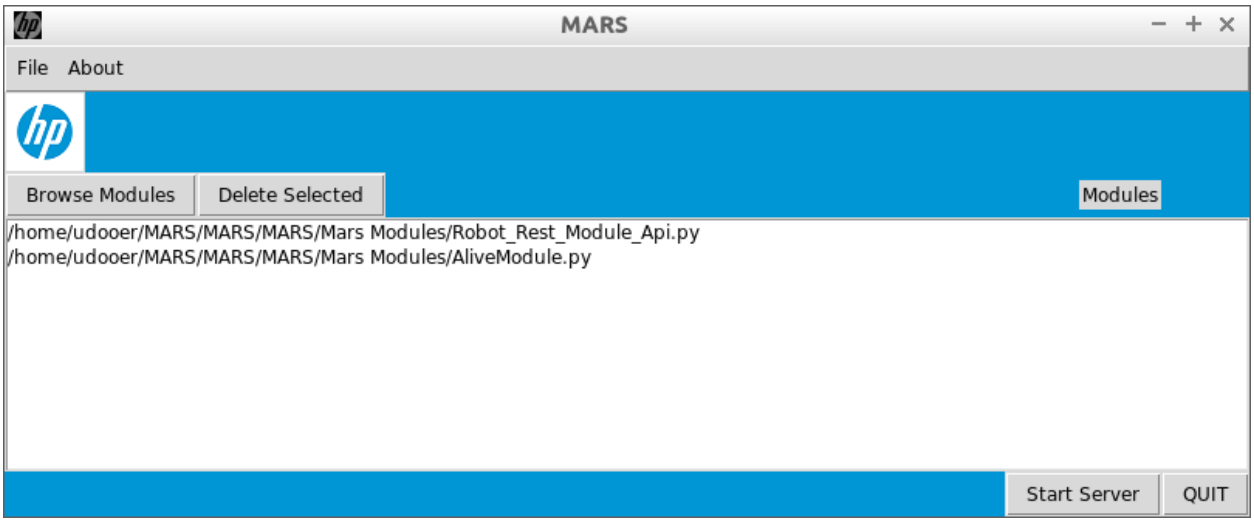


Figure 3. MARS GUI with Modules

When the user clicks Start Server, MARS parses the selected modules and makes them available via HTTP requests. Figure 4 shows the results of the operation and lists the individual classes that MARS exposed.

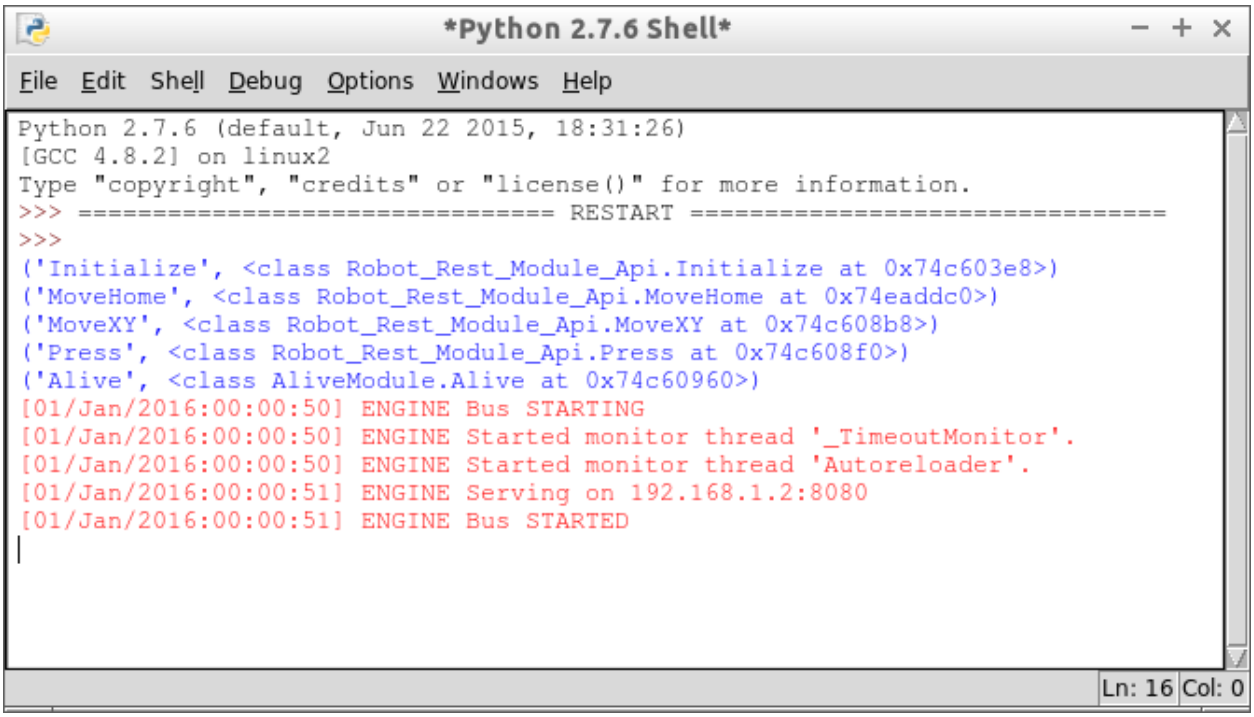


Figure 4. List of Exposed Classes

Altogether, MARS allows a user to bring functionality online in a modular manner. Additionally, a graphical interface is more intuitive to the average user than a command prompt. MARS is also able to save a list of modules to file so that future selection can be automated. Finally, for seamless operation, an administrator could start the server on power up using a Linux Daemon. A daemon is a process that runs in the background and is not manifest to the user; a daemon would enable an administrator to use the module list for automating startup and automating the exposure of the modules.

Development of Client

Development of the Client included the Dynamic Link Library, the demo application, and a test harness. The library is what provides the heart of the desired functionality; it also provides

an entry point through which test workflows can call physical button presses in the same manner that the workflows made calls for virtual button presses. The demo application was designed as a one-off application to demonstrate the capabilities of the library at the HP Intern Project Fair. The testing harness was developed to allow rapid testing of library calls at a command prompt.

Web Requests. Because the robot listens with an HTTP server, the client needs to communicate via HTTP. The student chose to communicate via an HTTP POST request to differentiate a call from a library and a call from a Web Browser, which uses HTTP GET requests. The library also uses basic authentication in order to keep out accidental hits to the robot URLs. Because the robot is located in an isolated lab environment, strict security was not a priority for the project. Figure 5 illustrates a call to the robot to initialize it for further use.

```
/// <summary>
/// Calibrates robot on startup
/// </summary>
/// <returns></returns>
public bool initialize()
{
    try
    {
        //calibrate robot
        string sURL;
        sURL = string.Format("http://{0}:{1}/api/Initialize", robotIP, roboPort);
        WebRequest wrPOSTURL;
        wrPOSTURL = WebRequest.Create(sURL);
        wrPOSTURL.Credentials = credentialCache;
        wrPOSTURL.ContentType = "application/x-www-form-urlencoded";
        wrPOSTURL.Method = "POST";
        wrPOSTURL.ContentLength = 0;
        wrPOSTURL.PreAuthenticate = true;
        //wrGETURL.Timeout = 3;
        Stream objStream;
        objStream = wrPOSTURL.GetResponse().GetResponseStream();
        objStream.Close();
        initialized = true;
        return true;
    }
    catch (WebException)
    {
        throw new WebException("An error occured communicating with the robot, ensure server is on and correct modules exposed.");
    }
}
```

Figure 5. Web Request for Initialization

Device Button Location. The device that was available for use during the project utilized WinForms, but the retrieval of properties required the use of web requests and another automation library. During this project, the student tried to discover the button locations

dynamically, but the location of a control is relative to the parent control. If the device were a local application, then the position could be determined by recursively adding together the ancestor coordinates. Due to perceived limitations in the original device automation library, the recursive solution did not appear viable.

Because the dynamic discovery of button locations was not available at the time of the project, the student manually discovered buttons relevant to the desired workflows and stored the button locations in a dictionary. The dictionary provides look-up capability in constant time and is, therefore, a good fit to rapidly search through buttons. Since the student has now implemented coordinate discovery in another utility, the potential to implement a more dynamic solution to the robotics library exists. Figure 6 illustrates an example dictionary with some sample buttons.

```
public readonly Dictionary<string, System.Drawing.Point> JediSignInFormDictionary800x600 = new Dictionary<string, System.Drawing.Point>()
{
    { "userField", new System.Drawing.Point(100, 170)},
    { "passwordField", new System.Drawing.Point(100, 255) },
    { "authField", new System.Drawing.Point(100,130) }
};
```

Figure 6. Dictionary

Coordinate Conversions. As is depicted in figure 7, the device that was available to the student had a screen with a resolution of 800 pixels wide by 600 pixels tall and an origin in the top-left. The robot had a field of 172 units wide by 142 units wide with an origin in the bottom right. Additionally, the device screen was smaller than the extents of the robot. Therefore, there was a need to implement a coordinate conversion function.

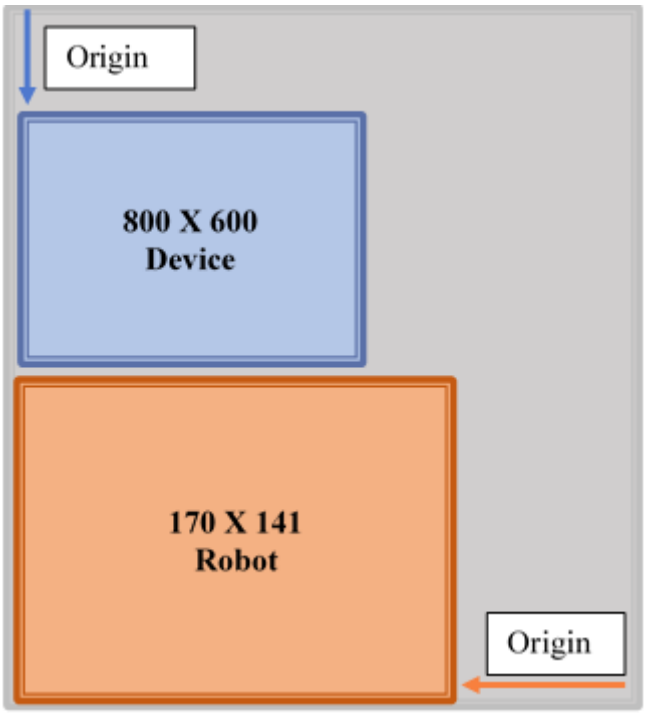


Figure 7. Differing Sizes and Origins

To map a range of values onto a new range of values, the student applied a version of Min-Max Normalization. Min-Max Normalization is a linear transformation of an existing range of values onto a new range. The equation for Min-Max Normalization is as follows in Equation 1 (Han, Kamber, Pei, 2012).

$$V_i' = \frac{V_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

(Eq. 1)

To account for the different origins, one can flip the coordinates by subtracting the old maximum value and taking the absolute value of the resulting value as follows in Equation 2.

$$V_i' = \left|\frac{V_i - max_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A\right|$$

(Eq. 2)

Subtracting the old max inverts the origin from that resembling quadrant IV to that resembling quadrant II. Since negative numbers are incompatible with the robot, the function

takes the absolute value of the result. Since the program knows the extents at initialization, it can perform some of the calculations ahead of time. The optimized value is shown in equation 3 and results in equation 4 on substitution.

$$Optimized_A = \frac{new\_max_A - new\_min_A}{max_A - min_A}$$

(Eq. 3)

$$V_i' = |(V_i - max_A)\ Optimized_A + new\_min_A|$$

(Eq. 4)

Offsets were used to establish artificial extents for the conversions and to compensate for the differing physical sizes. As is illustrated in figure 8, the function makes the robot proportions artificially smaller to ensure that all coordinates correspond to the screen.

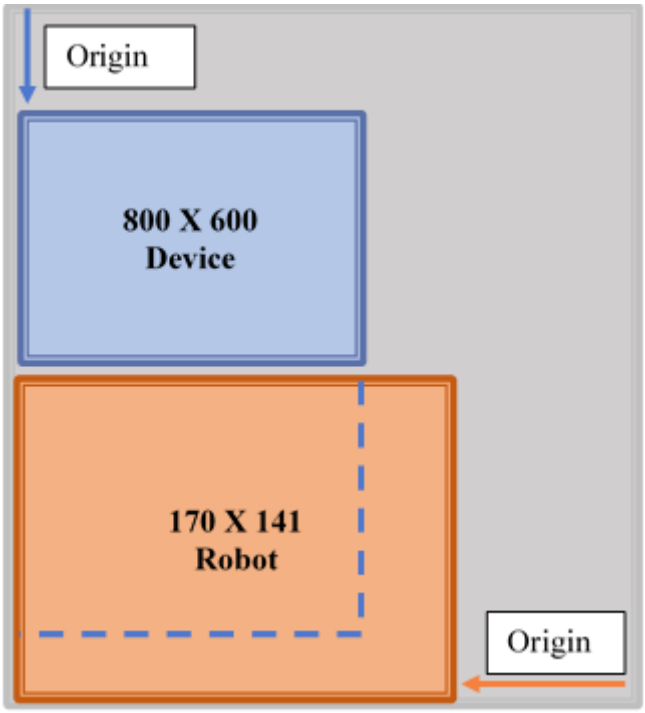


Figure 8. Offsets for Physical Size

Although the project dealt with screens of size 800 x 600, if the program could determine screen size at initialization then the function could convert the screen points to legitimate robot points.

Demo Application. The demo application is a WinForms based Graphical user interface. Most readers look at the top left of a document first and then scan from left to right; this region is part of the Power Zone (Markel, 2015). Because it is vital that a user find the most relevant information quickly, the student organized the controls inside of the Power Zone; therefore, one can easily find the most pertinent information to initialize and operate the demo. Figure 9 illustrates the Power Zone in the demo application.

Figure 9. Buttons in Power Zone

Furthermore, the student wanted the demonstration to allow interaction with a device. A screen shot was taken periodically of the device and displayed below the Power Zone. The user may double click anywhere on the image, and the robot will click the corresponding location on the actual device. Otherwise, the user may start the timer, and the robot will perform a

predetermined workflow every thirty seconds.

Documentation

The student created a user manual for the MARS server portion of the project. The user manual guides the reader through first time setup and how to configure a module for use. Additionally, the student created a poster for the HP Intern Project Fair which included several proprietary system diagrams. There is not a separate document for the robotics library as it is intended to be used in a very similar manner to the existing libraries.

Future Work

The student considered additional features or functionality that would make the robot much more useful. Compatibility with more devices and a shift to dynamic discovery of the buttons that the robot will push are the most prominent. The following two sections discuss the improvements more in depth.

More Devices

Currently, the library is only compatible with one type of printer. Further work to add more devices would make the library and robot much more applicable to the environment. Additionally, the student has developed a separate project to discover controls, buttons, or widgets of various devices. The library can be extended to work with multiple devices with minimal effort because the majority of the functionality for other devices already exists in a separate project.

Dynamic Control Discovery

One of the major hindrances to the project is the reliance on apriori knowledge of button locations. Adding the functionality to query the printer for button locations would make the project much more intuitive and user-friendly. All current devices support control discovery

through firmware calls. The student developed another project while at HP which performs control discovery but has not yet applied the functionality to the robotics project.

Coincidentally, after the termination of the project, the student discovered that the dot operator does in fact function in the library call. Therefore, a recursive or iterative solution was possible. Figure 10 illustrates a working solution that the student carried out in a separate project.

```
public int getOffsetIterativeParallel(string controlName)
{
    List<string> controls = new List<string>();
    int offset = 0;
    string parent = getProperty(controlName, "Parent");

    while (parent != "[null]")
    {
        controls.Add(controlName);
        parent = getProperty(controlName, "Parent");
        if (parent != "[null]")
            controlName = getProperty(controlName, "Parent.Name");
    }
    Parallel.ForEach(controls,
        (control) =>
        {
            try
            {
                //Entry Section
                int localOffset = Convert.ToInt32(getProperty(control, "Top"));
                //Critical Section
                propertiesLock.EnterWriteLock();
                offset += localOffset;
            }
            finally
            {
                //function absolutely must release lock regardless of successful completion
                propertiesLock.ExitWriteLock();
                //Remainder Section
            }
        });
    return offset;
}
```

Figure 10. Example of how to get correct coordinate of child control

Conclusion

The student completed analysis, development, support and debugging to accomplish a working project. Furthermore, the student met all the expressed requirements for the project. The library and robot can perform predetermined workflows with some human assistance. The student developed a C# library and Python server that interacts with a robot to run workflows on a device screen. The library and server provide the core functionality of the project, but there is more integration work, and there are many more useful features that a developer can create. Overall, the project was a success but is not expected to operate in widespread use.

References

CherryPy. (2016). Retrieved from http://cherrypy.org/

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: concepts and techniques*. Waltham, MA: Morgan Kaufmann.

Markel, M. H. (2015). *Technical communication*. Boston: Bedford/St. Martin's.

Appendix A: M.A.R.S.

Mars.py

```
#MARS.py
#Author: Timothy Mong
#Date of Last Revision: 07/08/16
#Purpose: Allows selection and exposure of correctly formatted python
modules in a RESTlike manner
#Libraries not included in Python: PIL, Tkinter, Cherrypy

import Tkinter
import tkFileDialog
import imp
from PIL import Image, ImageTk
from inspect import getmembers, isclass, ismodule
import cherrypy
import os
import argparse
import threading
import platform
#TODO --by someone,somewhere,sometime
#Expand functionality to include .pyc files to reduce repeated compiles


#Heart of the UI and functionality
class Application(Tkinter.Frame):

    _moduleList = list()
    _exposedClassList = list()
    _imported = False;
    _started = False;

    #attempts to rename class so that it has it's unique URL
    def duplicateResolution(self, className):
        duplicateCount = 1
        while (className in self._exposedClassList):
            className = className + str(duplicateCount)
            duplicateCount += 1
        return className

    def checkDuplicateClass(self, myClass, moduleName):
            myClass = moduleName + "/" + myClass

            if not (myClass) in self._exposedClassList:
                self._exposedClassList.append(myClass)
                return myClass
            else:
                revised = self.duplicateResolution(myClass)
                self._exposedClassList.append(revised)
                return revised
```

```
    def logExposedApi(self):
        logfile = open("ApiLog.txt", "w")
        for URL in self._exposedClassList:
            logfile.write("/api/" + URL + "\n")
        logfile.close()


    #Method: mountModules
    #Functionality: This method looks inside the selected modules to
expose the specific functionality of each on cherrypy
    #Parameters: self (Python does this automatically)
    def mountModules(self):
        for name in self._moduleList:

            moduleName = ""
            #for each module, get the classes therein
            for property, description in getmembers(name):
                if property == "__name__":
                    moduleName = description
            print moduleName
            classes = getmembers(name, isclass)

            #mount an instance of each class
            for aclass in classes:

                #strip off the consistent extra characters that inspection
returns
                #and make sure that there are not identical class names
                #Format: ('classname', <class classname.classname at
0xffffffff>)
                justName = str(aclass).split(",")
                justName = justName[0].strip("('")
                URL = self.checkDuplicateClass(justName, moduleName)
                cherrypy.tree.mount(getattr(name, justName)(), '/api/' +
URL,
                                    {'/':
                                    {'request.dispatch':
cherrypy.dispatch.MethodDispatcher()}  #This dispatcher expects REST
                                    })

        self.logExposedApi()
        #update your changes in the server.conf file to apply custom
hosting options
        #cherrypy.config.update(self.conf)    #security configs
        CONFIG = cherrypy.config.update('server.conf')  #user defined
configs
        cherrypy.engine.start()
        self.started = True
        cherrypy.engine.block()

    #Method: exportSelection
    #Functionality: export selected modules to file; This file may be used
to persist configuration details for future executions
```

```
    #Parameters: self (Python does this automatically)
    def exportSelection(self):
        try:
            options = {}
            options['filetypes'] = [('Module Config', '*.modcon')]
            options['title'] = 'Save Selection'
            savePath = tkFileDialog.asksaveasfilename(**options)
            if (os.path.exists(savePath)):
                selectionFile = open(savePath, "w")
                for index, listbox_filepath in
enumerate(self.moduleListbox.get(0, Tkinter.END)):
                    selectionFile.write(listbox_filepath + "\n")
                selectionFile.close()
                print "Selection saved to file at: " + savePath
            else:
                print "Either User canceled or pathname became invalid"
        except:
            print "Could not save selection."
            if (selectionFile):
                selectionFile.close()

    #Deprecated 8/5/16 -TSM
    #browses Import for methods...was used to load functions into a
listbox for user interactivity
    def browseImport(self, imported):
        #get and store class names to instantiate in server
        for className in getmembers(imported, isclass):
            print str(className)
            self._classList.append(className)

    #Method: runImportText
    #Functionality: Loads selection from textfile to server and checks for
duplicates
    #Parameters: self, filepath of the text (Python does this
automatically)
    def runImportText(self, loadPath):

        importList = list()
        try:
            selectionFile = open(loadPath, "r")
            selection = selectionFile.readline().strip("\n") #first entry
cannot be duplicate..so don't check
            if (selection):
                importList.append(selection)
                selection = selectionFile.readline().strip("\n") #Get
second entry and check for duplicates from here on out
                #While read operation succesful...not EOF
                while (selection):
                    found = False;
                    #check for duplicates...repeated linear search...worst
case asymptotic complexity O(n(n+1)/2)
                    for index, listbox_filepath in enumerate(importList):
                        if selection == listbox_filepath:
                            found = True
```

```
                        break
                    if found == False:
                        importList.append(selection)
                    #Get next line since read success is checked as loop
condition
                    selection = selectionFile.readline().strip("\n")
                #What is opened must be closed!
                selectionFile.close()
        except IOError as e:
            print "Error in importing text file: " + e.message
        finally:
            #Take pains for Resource release
            if (not selectionFile.closed):
                selectionFile.close()
        #For all of the modules that were not duplicates, parse the useful
info and load
        for index, listbox_filepath in enumerate(importList):
            self.parseAndLoad(index, listbox_filepath)
        #Indicate state
        self._imported = True

    #Method: runImport
    #Functionality: Imports modules that were added to listbox
    #Parameters: self (Python does this automatically)
    #Note: Does not check for duplicates since that is handled on entry of
selections
    def runImport(self):
        for index, listbox_filepath in enumerate(self.moduleListbox.get(0,
Tkinter.END)):
            self.parseAndLoad(index, listbox_filepath)

    #Method: runImport
    #Functionality: Imports modules that were added to listbox
    #Parameters: self (Python does this automatically)
    #Parameters: index -> the current location in the list of filepaths
    #Parameters: filepath -> a string filepath to a module
    def parseAndLoad(self, index, filepath):
                                                #Get Module name and
Extension
            split = os.path.split(filepath)     #os.path gives us platform
independent path splitting
            fileName = split[1]
            parsed = fileName.split('.')        #separate file name from
extension
            moduleName = parsed[0]              #name to left of extension
            extension = parsed[1]

            if (extension == "py"):             #check extension, refuse
if not python
                                                #add new module to list to
expose with CherryPy later
                                                #The pathnames are
retrieved from the gui
```

```
                self._moduleList.append(imp.load_source(moduleName,
filepath))
                #self.browseImport(self._moduleList[index])
            else:
                print moduleName + " is not an accepted python file.
Review your selection."

    #Method: openDialog
    #Functionality: Opens dialogue for to select python modules which are
loaded into a listbox
    #Parameters: self (Python does this automatically)
    #Notes: Use the options variable to filter different file extensions
    def openDialog(self):
        options = {}
        #TODO: Expand functionality to include .pyc files to reduce
repeated compiles
        #only browse python files
        options['filetypes'] = [('Module Files', '*Module*.py'), ('Python
Files', '.py')]
        options['title'] = 'Select Module'
        options['initialdir'] = 'Mars Modules'
        file_paths = tkFileDialog.askopenfilenames(**options)
        file_paths = root.tk.splitlist(file_paths)
        found = False;

        #check for duplicates...repeated linear search  O(n(n+1)/2) worst
case is no duplicates....go figure
        for number, myFile in enumerate(file_paths):
            #Ensure ascii for greater platform independence
            myFile = myFile.encode('ascii', 'ignore')
            found = False
            for index, listbox_filepath in
enumerate(self.moduleListbox.get(0, Tkinter.END)):
             #does not check if there are duplicates already in the
listbox but checks to see if adding the selection will create duplicate
                if found == False:
                    if myFile == listbox_filepath:
                        found = True
                        break
            if found == False:
                self.moduleListbox.insert(Tkinter.END, myFile)
        return

    #Method: deleteModule
    #Functionality: remove module from listbox
    #Parameters: self (Python does this automatically)
    def deleteModule(self):
        #remove item from module list
        try:
            index = self.moduleListbox.curselection()
            size =  self.moduleListbox.size()

            #check index validity
            if index:
```

```
                if (size != 0 and index[0] < size):  #make sure that there
is actually something to delete
                    self.moduleListbox.delete(index)
        except Exception:
            print "Previous index no longer valid"

    #Method: serverStart
    #Functionality: Start server exposing python modules that conform to
CherryPy REST
    #Parameters: self (Python does this automatically)
    def serverStart(self):

        if (self._imported == False):    #if the user hasn't initiated an
import separately
            self.runImport()
        if (not self._started):
            if (self._moduleList):
                self.mountModules()
            else:
                print "Server already started"

    #Method: loadConfig
    #Functionality: Opens text file and reads in the newline separated
pathnames of python modules.
    #Functionality: The pathnames are later used to import and mount the
modules
    #Parameters: self (Python does this automatically)
    def loadConfig(self):

        options = {}
        #TODO: Expand functionality to include .pyc files to reduce
repeated compiles
        #only browse python files
        options['filetypes'] = [('Module Config', '*.modcon')]
        options['title'] = 'Select config'
        loadPath = tkFileDialog.askopenfilename(**options)

        try:
            if (os.path.exists(loadPath)):
                selectionFile = open(loadPath, "r")
                selection = selectionFile.readline().strip("\n") #Get
second entry and check for duplicates from here on out
                while (selection):
                    found = False;
                    #check for duplicates...repeated linear search...worst
case asymptotic complexity O(n(n+1)/2)
                    for index, listbox_filepath in
enumerate(self.moduleListbox.get(0, Tkinter.END)):
                        if selection == listbox_filepath:
                            found = True
                            break
                    if found == False:
                        self.moduleListbox.insert(Tkinter.END, selection)
                    selection = selectionFile.readline().strip("\n")
```

```
                selectionFile.close()
            else:
                print "User Cancelled or Pathname became invalid"
        except IOError as e:
            print "Error in importing text file: " + e.message
            if (not selectionFile.closed):
                selectionFile.close()



    def displayAbout(self):
        ABOUT = """MARS.py
                Author: Timothy Mong
                Date of Last Revision: 7/7/16
                Purpose: Allows selection and exposure of python modules
in a RESTlike manner"""

        toplevel = Tkinter.Toplevel()
        label1 = Tkinter.Label(toplevel, text=ABOUT, height=0, width=75)
        label1.pack()
        toplevel.focus_force()


    #Method: createWidgets
    #Functionality: The buttons, textboxes, etc... are defined and added
to the layout manager
    #Parameters: self (Python does this automatically)
    def createWidgets(self):

        #Main menu bar
        self.mainMenu = Tkinter.Menu(self)


        #File dropDown
        menu = Tkinter.Menu(self.mainMenu, tearoff=0)
        self.mainMenu.add_cascade(label="File", menu=menu)
        menu.add_command(label = "Load Config", command = self.loadConfig)
        menu.add_command(label = "Save Config", command =
self.exportSelection)
        menu.add_command(label = "Quit", command = self.quit)


        #About
        menu = Tkinter.Menu(self.mainMenu, tearoff=0)
        self.mainMenu.add_command(label="About", command =
self.displayAbout)
        root.config(menu = self.mainMenu)

        self.topButtonsPanel = Tkinter.Frame(self)
        self.topButtonsPanel.grid(row = 3, column = 0, sticky =Tkinter.W)

        #Browse button
        self.findFiles = Tkinter.Button(self.topButtonsPanel, text =
"Browse Modules", command = self.openDialog)
```

```
        self.findFiles.grid(row = 3, column = 0, sticky =Tkinter.W +
Tkinter.N+Tkinter.S+Tkinter.E)

        #Delete Button
        self.delete = Tkinter.Button(self.topButtonsPanel, text = "Delete
Selected", command =  self.deleteModule)
        self.delete.grid(row = 3, column = 1, sticky =Tkinter.W +
Tkinter.N+Tkinter.S+Tkinter.E)

        #Quit button
        self.QUIT = Tkinter.Button(self, text = "QUIT", command =
self.stopServer)
        self.QUIT.grid(row = 15, column = 3, sticky = Tkinter.E +
Tkinter.W)

        #HPI Logo
        try:
            self.logo = Image.open("HP_logo.png")
            self.photo = ImageTk.PhotoImage(self.logo)
            self.logoLabel = Tkinter.Label(self, image = self.photo)
            self.logoLabel.image = self.photo
            self.logoLabel.grid(row = 0, column = 0, sticky = Tkinter.W)
        except Exception:
            print "Couldn't load logo. To fix, check that the logo is in
the project folder."


        #start server button
        ServerThread = threading.Thread(target = self.serverStart)
        ServerThread.daemon = True
        self.startServer = Tkinter.Button(self, text = "Start Server",
command = lambda: ServerThread.start())
        self.startServer.grid(row = 15, column = 2, sticky = Tkinter.W +
Tkinter.E)


        #Module Listbox Label
        self.methodLabel = Tkinter.Label(self, text = 'Modules')
        self.methodLabel.grid(row = 3, column = 2, sticky = Tkinter.E)

        #Module Listbox
        self.moduleListbox = Tkinter.Listbox(self, width = 100)
        self.moduleListbox.grid(row = 4, column = 0, rowspan = 4,
columnspan = 4  ,sticky = Tkinter.W + Tkinter.E+ Tkinter.N+ Tkinter.S)


    #Method: stopServer
    #Functionality: Uses the cherrypy engine to stop the server and uses
quit to tear down the gui
    #Parameters: self (Python does this automatically)
    def stopServer(self):
        if cherrypy.engine.state.name != "STOPPED":
            cherrypy.engine.stop()
        self.quit()
```

```
    #Method: setup
    #Functionality: Sets basic gui configurations such as background
color, layout manager, and calls the createWidget method
    #Parameters: self (Python does this automatically)
    def setup(self, master = None):
        Tkinter.Frame.__init__(self, master)
        #Change protocol to make sure server thread is stopped before
dumping gui
        master.protocol("WM_DELETE_WINDOW", self.stopServer)
        master.resizable(0,0)
        self.config(background = '#0096d6') #HP Blue Accent 1 Red = 0
0x00, Green = 150 0x96, Blue = 214 0xd6
        master.title("MARS")


        try:
            if (platform.system() == 'Windows'):
                master.iconbitmap(default = 'HPLOGO.ico')
            else:
                master.iconbitmap('@HPLOGO.xbm')
        except Exception as e:
            print "No Icon in folder or is wrong format"
        self.grid(sticky = Tkinter.W  + Tkinter.S + Tkinter.N + Tkinter.E)
        self.grid_columnconfigure(0, weight=1)
        self.createWidgets()


########################## MAIN
################################################
#functions as main
if __name__ == '__main__': #system variable

    parser = argparse.ArgumentParser(description='Check for filepath')
    parser.add_argument('--pathname', help = "Enter pathname of desired
text file")
    args = parser.parse_args()

    try:

        if args.pathname:
            #if filename is passed in, start without gui
            if (os.path.exists(args.pathname)): #if file exists
                app = Application()
                app.runImportText(args.pathname)
                app.serverStart()
            #start with gui
            else:                                   #if file does not exist as
entered
                root = Tkinter.Tk()
                app = Application()
                app.setup(master = root)
                app.mainloop()
                root.destroy()
```

```
        else: #no arguments
            root = Tkinter.Tk()
            app = Application()
            app.setup(master = root)
            app.mainloop()
            root.destroy()

    except ImportError as ie:
        print "Something went wrong with importing the list of modules and
mounting to the server"
        print ie.message
        print ie.args

    except Exception as e:
        print "Server could not start. Ensure that modules conform to
CherryPY REST architecture"
        print e.message
        print e.args
```

Robot_Rest_Module_Api.py

```
import cherrypy
import os
import UI_Interface

ui = UI_Interface.UI_Interface()


class MoveHome:
    exposed = True
    def GET(self):
        if (ui.moveToXY(0,)):
            return "Move home complete"
        else:
            return "Move failed, recommend recalibrate"
    def POST(self):
        if (ui.moveToXY(0,)):
            return "Move home complete"
        else:
            return "Move failed, recommend recalibrate"

class Initialize:
    exposed = True

    def GET(self):
        if ui.calibrate():
            return "Calibration Succesful"
        else:
            return "Calibration failed, please recalibrate"
    def POST(self):
        if ui.calibrate():
            return "Calibration Succesful"
        else:
```

```
            return "Calibration failed, please recalibrate"

class MoveXY:
    exposed = True

    def GET(self, x=0,  y=0):
        x =  int(x)
        y =  int(y)


        if (x <= 170 and x >= 0) and (y <= 141 and y >= 0):

            if ui.moveToXY(x,y):
                return "Move Complete"
            else:
                return "Move failed, recommend recalibrate"
        else:
            return "Invalid Coordinates"
    def POST(self, x=0,  y=0):
        x =  int(x)
        y =  int(y)


        if (x <= 170 and x >= 0) and (y <= 141 and y >= 0):

            if ui.moveToXY(x,y):
                return "Move Complete"
            else:
                return "Move failed, recommend recalibrate"
        else:
            return "Invalid Coordinates"

class Press:
    exposed  =  True

    def GET(self, z=5,  duration=1.0):
        z= int(z)
        duration = float(duration)

        if (z <= 15 and z >= 0) and (duration <= 1.0 and duration >= 0.0):
            if ui.press(z,duration):
                return "Press Complete"
            else:
                return "Button Press failed"
        else:
            return "Invalid Coordinates"

    def POST(self, z=5,  duration=1.0):
        z= int(z)
        duration = float(duration)

        if (z <= 15 and z >= 0) and (duration <= 1.0 and duration >= 0.0):
            if ui.press(z,duration):
                return "Press Complete"
```

```
            else:
                return "Button Press failed"
        else:
            return "Invalid Coordinates"

if __name__ == '__main__':

    conf = {
            'global':{
                   'server.socket_host': '15.41.185.99',
                    'server.socket_port': 8080,
                    },
   }
    cherrypy.tree.mount(
       Initialize(), '/Initialize',
        {'/':
           {'request.dispatch': cherrypy.dispatch.MethodDispatcher()}
        }
    )
    cherrypy.config.update('server.conf')
    cherrypy.engine.start()
    cherrypy.engine.block()
```

AliveModule.py

```
#See https://cherrypy.readthedocs.io/en/3.3.0/tutorial/REST.html

import cherrypy


class Alive:

    exposed = True

    def GET(self):
        return "alive"

    def POST(self):
        return "alive"

if __name__ == '__main__':

    cherrypy.tree.mount(
        Lander(), '/api/Lander',
        {'/':
            {'request.dispatch': cherrypy.dispatch.MethodDispatcher()}
        }
    )
    cherrypy.engine.start()
    cherrypy.engine.block()
```

SecurityModule.py

```
#Security Module to implement authentication and other niceties that might
be wanted later

import cherrypy
from cherrypy.lib import auth_basic

USERS = {'REDACTED': 'REDACTED'}
class Security():

    def validate_password(realm, username, password):
        if username in USERS and USERS[username] == password:
           return True
        return False
    conf = {
        'tools.auth_basic.on': True,
        'tools.auth_basic.realm': 'localhost',
        'tools.auth_basic.checkpassword': validate_password}
    cherrypy.config.update(conf)
    #exposed = True
    #def GET(self):
    #   return ""

class Alive():
    exposed = True;

    def GET():
        return "hello"
```

Example Server Configuration File: server.conf

```
[global]
server.socket_host:"192.168.0.2"
server.socket_port: 8080
```

Example Module Configuration File: SelectedModules.txt

```
C:/Users/REDACTED/Desktop/robotics/Robotics/MARS/MARS/Mars
Modules/AliveModule.py
C:/Users/REDACTED/Desktop/robotics/Robotics/MARS/MARS/Mars
Modules/SecurityModule.py
```

MARS Documentation Example CherryPy Module: Lander.py

```
#See https://cherrypy.readthedocs.io/en/3.3.0/tutorial/REST.html

import cherrypy

thrusters = {
    '1': {
        'title': 'Anterior Guidance',
        'state': 'Firing'
    },

    '2': {
        'title': 'Posterior Guidance',
        'state': 'Firing'
    },

    '3': {
        'title': 'Inferior Main',
        'state': 'Bingo Fuel'
    }
}

class Lander:

    exposed = True

    def GET(self, id=None):

        if id == None:
            return('Here are all the thrusters we have: %s' % thrusters)
        elif id in thrusters:
            thruster = thrusters[id]
            return('Thruster %s is the %s thruster, and it is currently
%s' % (id, thruster['title'], thruster['state']))
        else:
            return('No thruster with the ID %s :-(' % id)


if __name__ == '__main__':

    cherrypy.tree.mount(
        Lander(), '/api/Lander',
        {'/':
            {'request.dispatch': cherrypy.dispatch.MethodDispatcher()}
        }
    )

    cherrypy.engine.start()
    cherrypy.engine.block()
```

MARS Documentation Example CherryPy Module: Orbiter.py

```
#See https://cherrypy.readthedocs.io/en/3.3.0/tutorial/REST.html

import cherrypy


class Orbiter:

    exposed = True

    def GET(self, id=None):

        return "No Contact"



if __name__ == '__main__':

    cherrypy.tree.mount(
        Orbiter(), '/api/Orbiter',
        {'/':
            {'request.dispatch': cherrypy.dispatch.MethodDispatcher()}
        }
    )

    cherrypy.engine.start()
    cherrypy.engine.block()
```

MARS Documentation Example CherryPy Module: SatCom.py

```
#See https://cherrypy.readthedocs.io/en/3.3.0/tutorial/REST.html

import cherrypy

links = {
    '1': {
        'title': 'EarthComs',
        'connection': 'Active'
    },

    '2': {
        'title': 'LanderComs',
        'connection': 'Disabled'
    },

    '3': {
        'title': 'Red Net',
        'connection': 'Active'
    }
}

class SatCom:
```

```
    exposed = True

    def GET(self, id=None):

        if id == None:
            return('Here are all the channels we have: %s' % links)
        elif id in links:
            channel = links[id]
            return('Com channel %s is called %s, and is %s' % (id,
channel['title'], channel['connection']))
        else:
            return('No channel with the ID %s :-(' % id)



if __name__ == '__main__':

    cherrypy.tree.mount(
        SatCom(), '/api/SatCom',
        {'/':
            {'request.dispatch': cherrypy.dispatch.MethodDispatcher()}
        }
    )

    cherrypy.engine.start()
    cherrypy.engine.block()
```

MARS Documentation: MARS Documentation.pdf

{HPI LOGO}

Modular Architecture Robot Server (MARS)

Reliable Deployment Lab

Intro:

MARS is a server framework to expose resources and methods in a Restlike manner. The *cherrypy* engine provides the majority of the server functionality while the *imp* library allows us to dynamically import REST APIs (python modules that conform to cherrypy REST examples). The MARS project allows functionality to be added or removed from the server with minimal effect on the other independent modules. The goal is to increase productivity by allowing a developer to bring online bits of functionality although the whole program might

not yet be complete. There are two main ways to run this program. The first method utilizes a gui to browse and select modules. The second requires a pathname to a text file that contains the pathnames to the desired modules separated by newlines. The following documentation shows how to run the setup in either of the two aformentioned manners.

Outside Libraries Utilized:

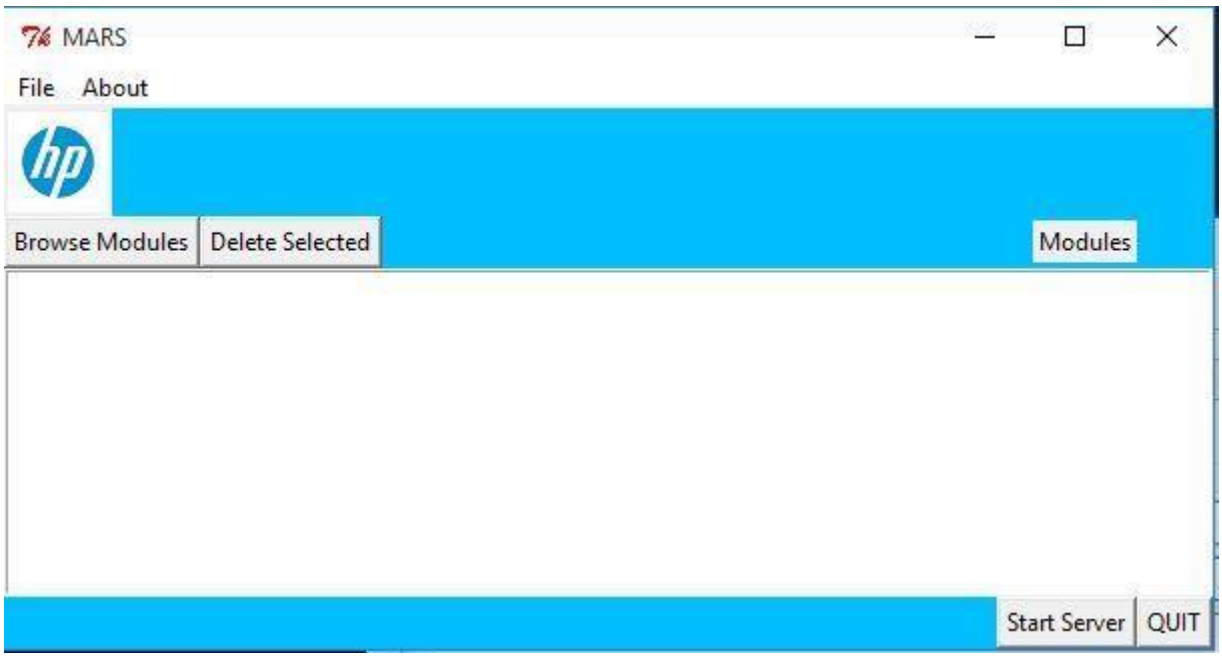
- Cherrypy
- Tkinter
- PIL

Python Libraries Utilized:

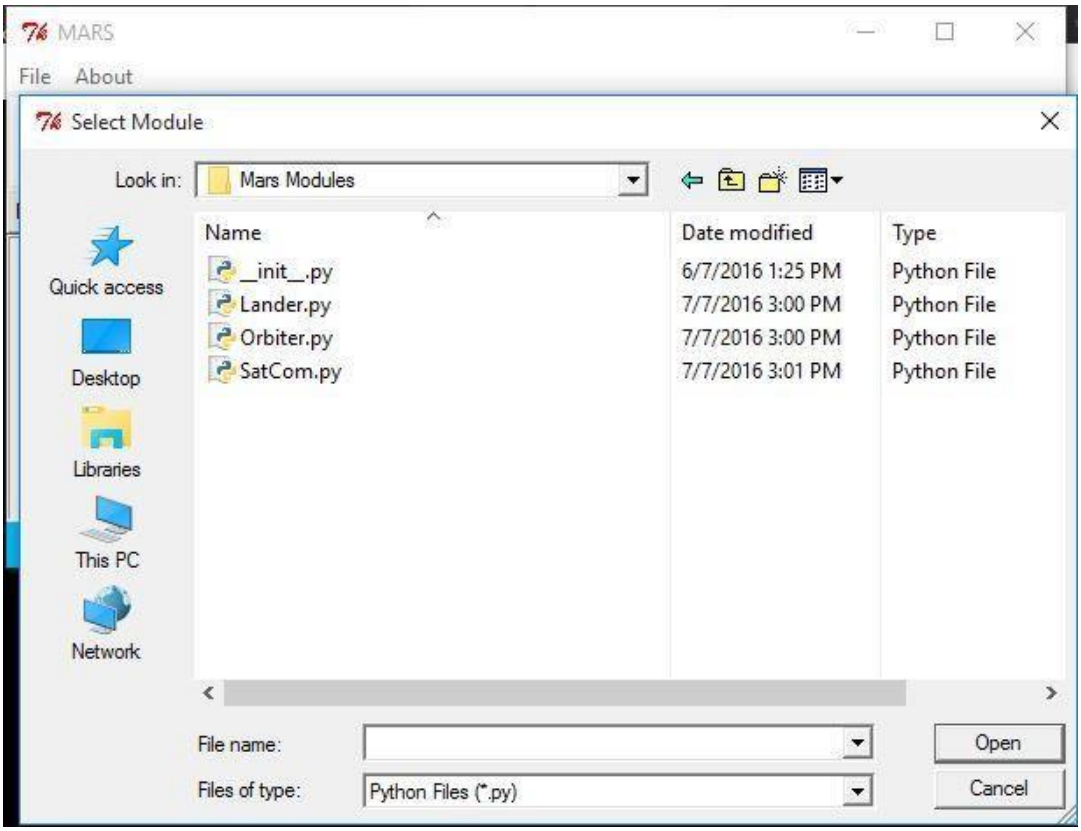
- imp
- inspect
- os
- argparse

Start Via GUI:

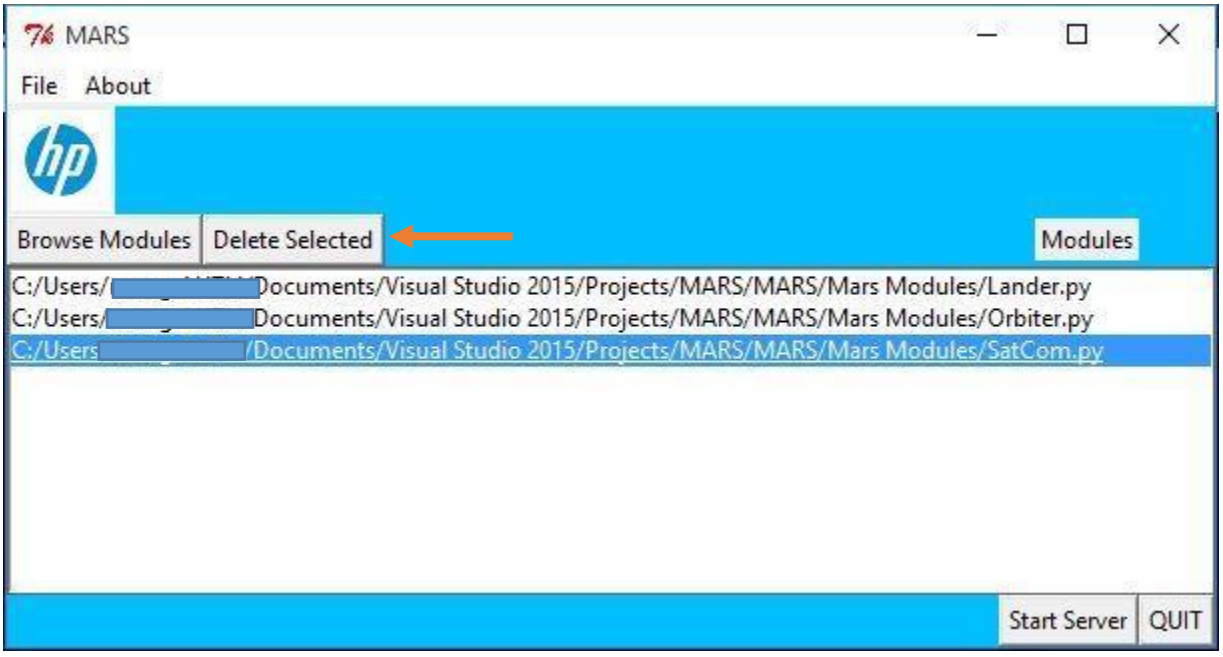
1. Run the python module without any command line arguments. For example, **"python MARS.py"**
2. The home screen should look like this:



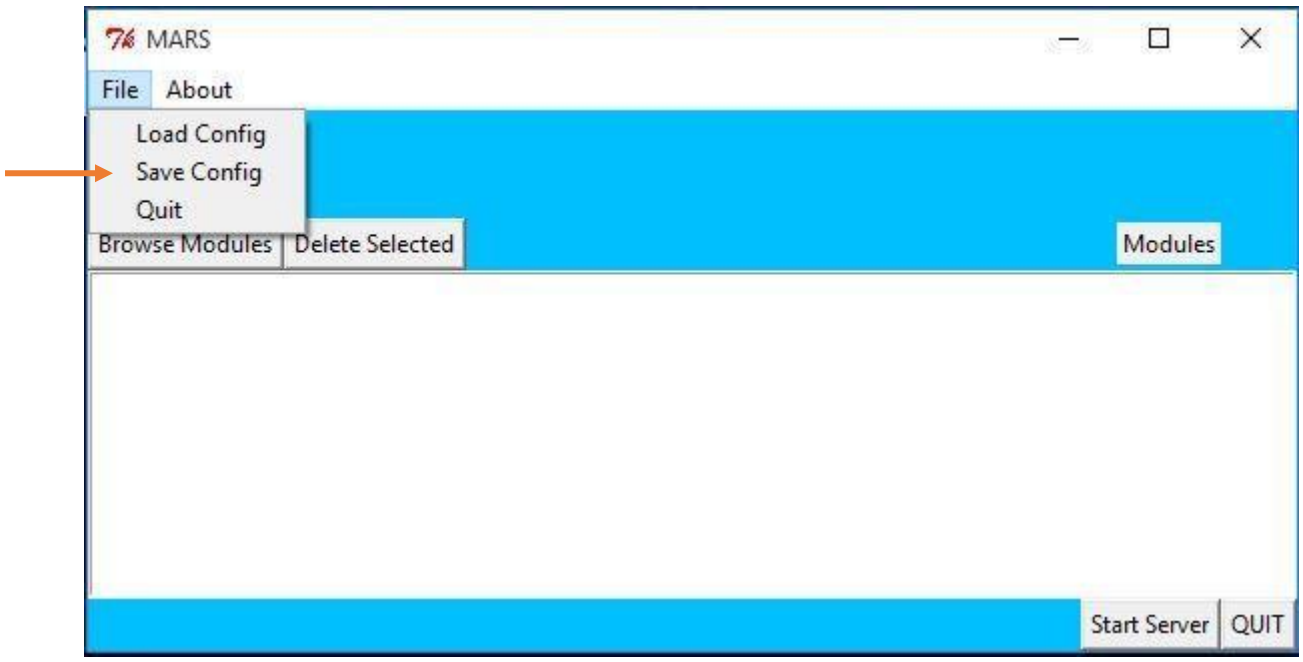
3. On first time setup, it is likely that you will want to select your modules manually.



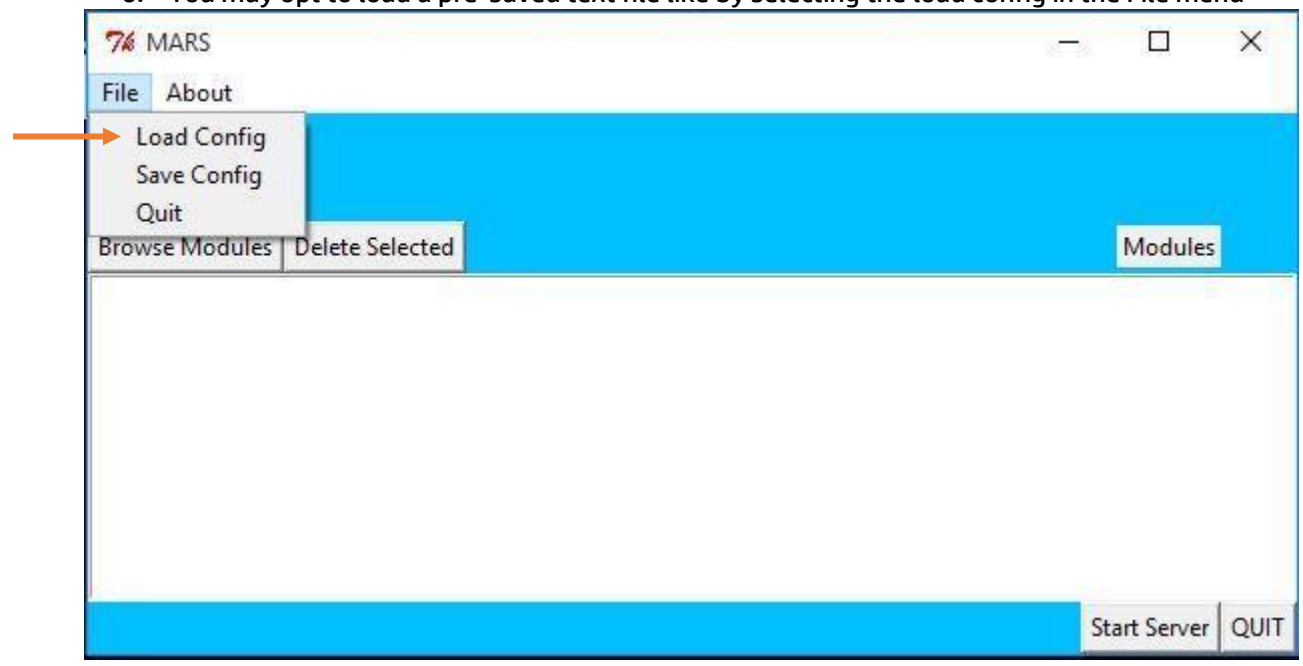
4. Sometimes you might select the wrong thing and need to delete a module. To do this, highlight the module and click delete.



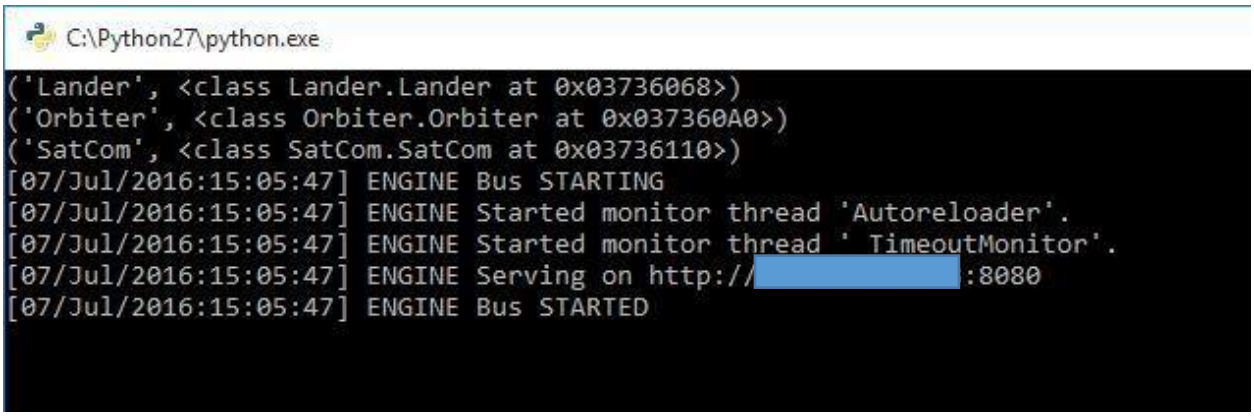
5. If you want, you may save your selected modules to file for future use.



6. You may opt to load a pre-saved text file like by selecting the load config in the File menu

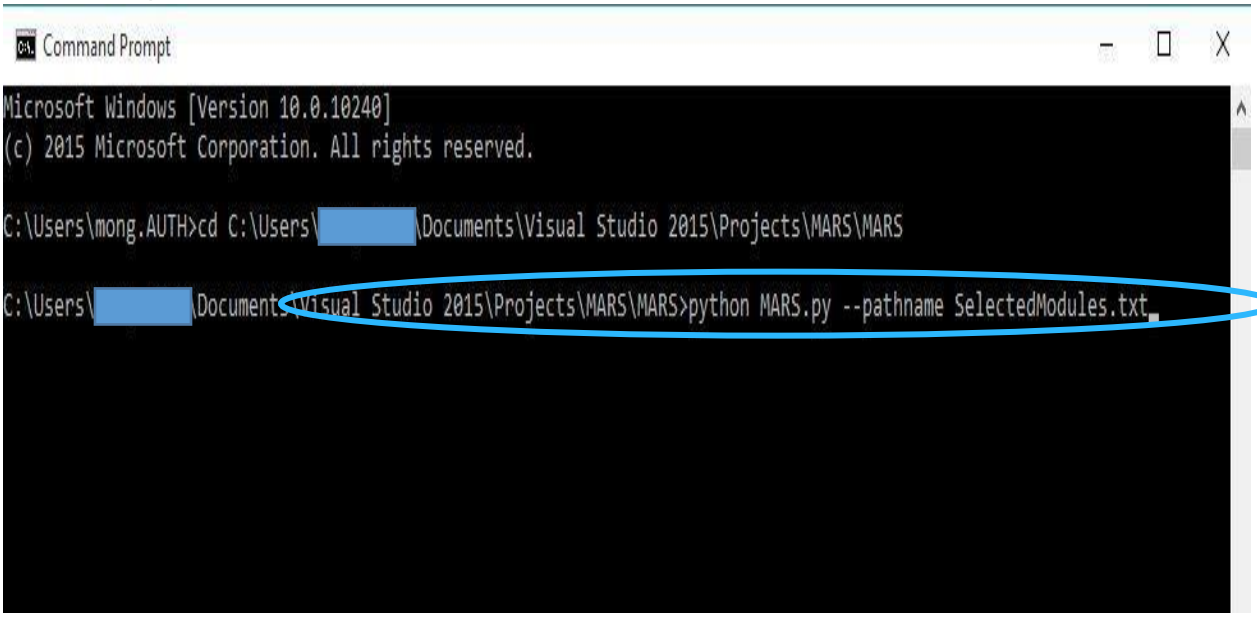


7. When you have selected the modules you want, click **Start Server.**



Start Via Command Line Argument:

1. To start with preexisting text file (useful for startup scripts). Simply provide the extra command line parameters.



Coding Requirements:

1. Modules must conform to the cherrypy rest examples.

The examples follow the general form: import

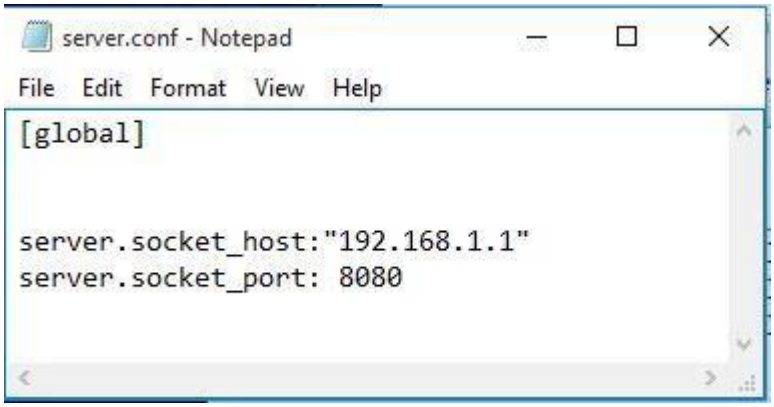
cherrypy

```
import yourextramodules #(an api that your GET can call into, for example) class

myFunction:

        exposed = True

        def GET(your parameters here):
                #your code here
        def POST:
#your code here
def PUT:
#your code here
def DELETE:
#your code here
```

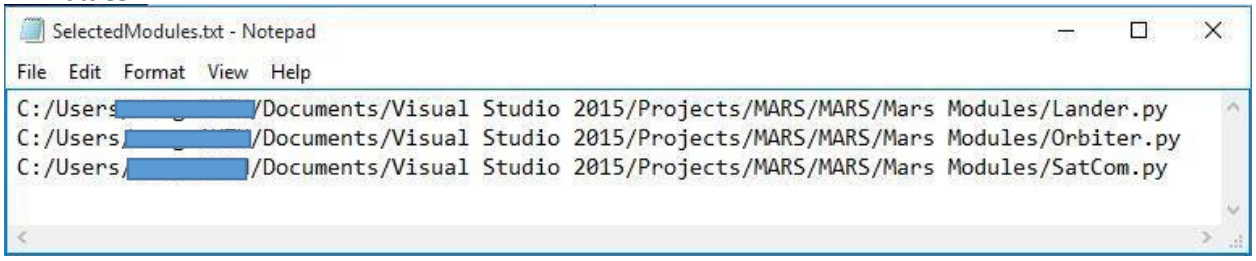
see https://cherrypy.readthedocs.io/en/3.3.0/tutorial/REST.html

Configuration Requirements:

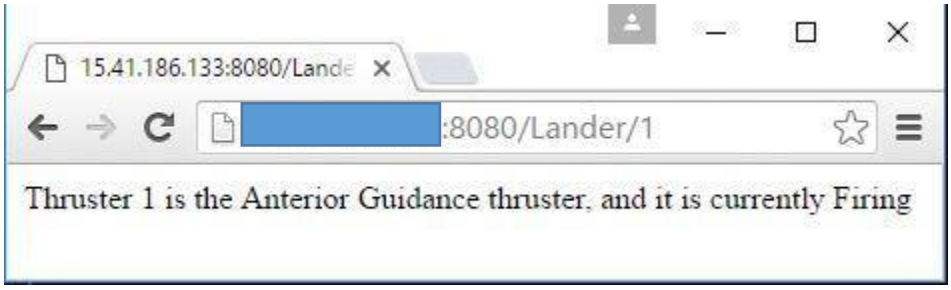
1. Make any necessary changes to server configuration via the *server.conf* file. There are numerous examples on the cherrypy website.



2. Save and load modules using the text file "SelectedModules.txt" using newline separated values.



3. Verify success with your browser


15.41.186.133:8080/Lande
:8080/Lander/1
Thruster 1 is the Anterior Guidance thruster, and it is currently Firing

Appendix B: Robot Library

Robot.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using HP.DeviceAutomation.Jedi;
using HP.ScalableTest.Data.AssetInventory.Model;
using System.Threading;
using System.Threading.Tasks;
namespace RobotLibrary
{
    public interface IRobot
    {
        bool setupRobot(string ipaddress);
        IEnumerable<string> getStatus();
        bool pressXY(int x, int y);
        bool scanToDeviceMemory(string filename);
        bool retrieveFromDeviceMemory(string whichDocument, bool delete);
        bool makeCopy(string numberOfCopies);
        bool hpacPullPrint(string user, string password);
        bool hpacPullPrintIRM(string auth);
    }

    public class JediWindjammerRobot : IRobot, IDisposable
    {
        private string robotIP = String.Empty;
        private string deviceIP = String.Empty;
        private string roboPort = "8080";
        private JediWindjammerDevice device = null;
        private bool disposed = false;
        private bool initialized = false;
        private double[] lastMove = { 0, 0 };
        private System.Net.CredentialCache credentialCache = new
System.Net.CredentialCache();
        public Cartographer deviceMap = null;
        private ReaderWriterLockSlim propertiesLock;


        /// <summary>
        /// Checks status of robot, alive means that the robot is properly
configured and listening
        /// </summary>
        /// <returns></returns>
        public IEnumerable<string> getStatus()
        {
            List<string> status = new List<string>();
            status.Add(string.Format("Robot IP: {0}", robotIP));
            status.Add(string.Format("Initialized: {0}", initialized));
            status.Add(string.Format("Is Alive? {0}", isAlive()));
```

```
            status.Add(string.Format("Last Point: " + "({0},{1})",
lastMove[0], lastMove[1]));
            return status;

        }
        /// <summary>
        /// allows a program to specify the coordinates that the robot
should press
        /// </summary>
        /// <param name="x"></param>
        /// <param name="y"></param>
        /// <returns></returns>
        public bool pressXY(int x, int y)
        {
            pressXY800x600(new System.Drawing.Point { X = x, Y = y });
            return true;
        }

        /// <summary>
        /// Constructs an object that can interact with a robot based on
the passed device and robot data
        /// </summary>
        /// <param name="deviceAddress"></param>
        /// <param name="robotAddress"></param>
        public JediWindjammerRobot(JediWindjammerDevice wantedDevice,
string robotAddress)//, Robot robotData)
        {
            device = wantedDevice;
            robotIP = robotAddress;
            //TODO add device UI size and instantiate cartographer
apporopriately
            deviceMap = new Cartographer(800, 600);
            credentialCache.Add(
                new System.Uri(string.Format("http://{0}:8080/",
robotIP)),
                "Basic",
                new System.Net.NetworkCredential("robot", "NotARealPass")
            );
            if (!setupRobot(robotIP))
            {
                throw new Exception("Robot is offline or Alive module not
activated.");
            }
            else
            {
                Console.WriteLine("Robot and Device are ready for use\n---
------------------------------");
            }
        }

        /// <summary>
        /// Constructs an object that can interact with a robot based on
the passed device and robot data
        /// </summary>
```

```
        /// <param name="deviceAddress"></param>
        /// <param name="robotAddress"></param>
        public JediWindjammerRobot(JediWindjammerDevice wantedDevice,
Robot robotData)
        {
            device = wantedDevice;
            robotIP = robotData.Address;
            //could add more robot extents later if discovered that robot
extents vary
            //TODO add device UI size and instantiate cartographer
apporopriately
            deviceMap = new Cartographer(800, 600);
            credentialCache.Add(
                new System.Uri(string.Format("http://{0}:{1}/",
robotData.Address, "8080")),
                "Basic",
            new System.Net.NetworkCredential("robot", "NotaRealPAss")
            );
            if (!setupRobot(robotIP))
            {
                throw new Exception("Robot is offline or Alive module not
activated.");
            }
            else
            {
                Console.WriteLine("Robot and Device are ready for use\n---
------------------------------");
            }
        }
        /// <summary>
        ///
        /// </summary>
        public void Dispose()
        {
            if (!disposed)
            {
                if (device != null)
                {
                    device.Dispose();
                }
                if (deviceMap != null)
                {
                }
            }
            disposed = true;
        }

        /// <summary>
        /// Sets up robot for use and checks robot status
        /// </summary>
        /// <param name="ipAddress"></param>
        /// <returns>
        /// boolean
        /// </returns>
```

```
        public bool setupRobot(string ipAddress)
        {
            robotIP = ipAddress;
            return isAlive() ? true : false;
        }

        public bool deviceStub(string buttonID)
        {
            if (device != null)
            {


Console.WriteLine(device.ControlPanel.GetProperty<int>(buttonID, "Top"));

Console.WriteLine(device.ControlPanel.GetProperty<int>(buttonID,
"Bottom"));

Console.WriteLine(device.ControlPanel.GetProperty<int>("mHeader", "Top"));

Console.WriteLine(device.ControlPanel.GetProperty<int>("mHeader",
"Bottom"));

Console.WriteLine(device.ControlPanel.GetProperty<int>("SaveToDeviceMemory
App", "Top"));

Console.WriteLine(device.ControlPanel.GetProperty<int>("SaveToDeviceMemory
App", "Bottom"));


                //HP.DeviceAutomation.JavaScriptEngine engine = new
HP.DeviceAutomation.JavaScriptEngine(device.ControlPanel);

                //Console.WriteLine(JavaScriptUtil.RetrieveHtml(engine));

//Console.WriteLine(JavaScriptUtil.RetrieveBoundingAreaByElementId(engine,
buttonID));
                Console.ReadLine();
                //PadawanLearner.GetJediFormOffset();
            }
            return true;
        }

        /// <summary>
        /// Checks Robot Status
        /// </summary>
        /// <returns>
        /// boolean
        /// </returns>
        public bool isAlive()
        {
            string sURL = string.Format("http://{0}:{1}/api/Alive",
robotIP, roboPort);

            WebRequest wrPOSTURL = WebRequest.Create(sURL);
```

```
            wrPOSTURL.ContentType = "application/x-www-form-urlencoded";
            wrPOSTURL.Method = "POST";
            wrPOSTURL.ContentLength = 0;
            wrPOSTURL.Credentials = credentialCache;
            wrPOSTURL.PreAuthenticate = true;
            wrPOSTURL.Timeout = 2000;
            try
            {
                using (Stream objStream =
wrPOSTURL.GetResponse().GetResponseStream())
                {
                    string response = new
StreamReader(objStream).ReadToEnd();
                    if (response == "alive")
                    {
                        return true;
                    }
                    else
                    {
                        return false;
                    }
                }
            }
            catch (WebException)
            {
                return false;
            }
        }


        /// <summary>
        /// Save to device memory
        /// </summary>
        /// <param name="fileName"></param>
        /// <returns></returns>
        public bool scanToDeviceMemory(string fileName)
        {
            if (device != null)
            {
                device.PowerManagement.Wake();

moveAndPress(deviceMap.JediMainDictionary800x600["SaveToDeviceMemoryApp"])
;

                for (int x = 2; x > 0 &&
!device.ControlPanel.WaitForForm("SaveToDeviceMainForm"); x--) //retry
twice if form load fails
                {

moveAndPress(deviceMap.JediScanToDeviceDictionary800x600["returnHome"]);
//make sure not on some other app...button in common location across forms

moveAndPress(deviceMap.JediMainDictionary800x600["SaveToDeviceMemoryApp"])
;
```

```
                }


moveAndPress(deviceMap.JediScanToDeviceDictionary800x600["firstfolder"]);

moveAndPress(deviceMap.JediScanToDeviceDictionary800x600["jobname"]);
                for (int x = 2; x > 0 &&
!device.ControlPanel.WaitForForm("SaveToDeviceKeyboardForm"); x--) //retry
twice if form load fails)
                {

moveAndPress(deviceMap.JediScanToDeviceDictionary800x600["jobname"]);
                }

                if (device.ControlPanel.CurrentForm() ==
"SaveToDeviceKeyboardForm")    //if keyboard finally came up enter
password
                {
                    foreach (char letter in fileName)
                    {
                        pressKeyboard(letter.ToString());
                    }

moveAndPress(deviceMap.JediKeyboardDictionary800x600["mButtonOK"]);
                }
                else
                {
                    return false;
                }

                if
(device.ControlPanel.WaitForForm("SaveToDeviceMainForm"))
                {

moveAndPress(deviceMap.JediScanToDeviceDictionary800x600["startscan"]);
                }
                for (int x = 2; x > 0 &&
!device.ControlPanel.WaitForForm("BaseJobStartPopup"); x--) //retry twice
if form load fails)
                {

moveAndPress(deviceMap.JediScanToDeviceDictionary800x600["startscan"]);
                }
                if (device.ControlPanel.CurrentForm() ==
"BaseJobStartPopup")
                {

moveAndPress(deviceMap.JediBaseJobStartPopupDictionary800x600["ok"]);

device.ControlPanel.WaitForForm("SaveToDeviceMainForm");

moveAndPress(deviceMap.JediScanToDeviceDictionary800x600["returnHome"]);
                    return true;
                }
```

```
                else
                {
                    return false;
                }
            }
            else
            {
                throw new Exception("Printer/Device not instantiated");
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="whichDoc"></param>
        /// <returns></returns>
        public bool retrieveFromDeviceMemory(string whichDoc, bool delete)
        {
            if (device != null)
            {
                device.PowerManagement.Wake();

moveAndPress(deviceMap.JediMainDictionary800x600["OpenFromDeviceMemoryApp"
]);


device.ControlPanel.WaitForForm("OpenFromDeviceMemoryMainForm");


moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600["firstfolde
r"]);
                //'all' or 'firstDoc'
                if
(deviceMap.JediRetrieveFromDeviceDictionary800x600.ContainsKey(whichDoc))
                {

moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600["all"]);

moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600[whichDoc]);

device.ControlPanel.WaitForForm("OpenFromDeviceMemoryMainForm");

moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600["startretri
eve"]);
                    if (whichDoc == "firstDoc")
                    {

device.ControlPanel.WaitForForm("BaseJobStartPopup");

moveAndPress(deviceMap.JediBaseJobStartPopupDictionary800x600["ok"]);
                    }
                    else
                    {
                        device.ControlPanel.WaitForForm("HPMessageBox");
```

```
moveAndPress(deviceMap.JediHPMessageBoxDictionary800x600["ok"]);
                    }

                    if (delete)
                    {

device.ControlPanel.WaitForForm("OpenFromDeviceMemoryMainForm");

moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600["firstfolde
r"]);

moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600[whichDoc]);

moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600["delete"]);

device.ControlPanel.WaitForForm("TwoButtonMessageBox");

moveAndPress(deviceMap.JediTwoButtonMessageBoxDictionary800x600["ok"]);

device.ControlPanel.WaitForForm("OpenFromDeviceMemoryMainForm");

moveAndPress(deviceMap.JediRetrieveFromDeviceDictionary800x600["returnHome
"]);
                    }
                    return true;
                }
                else
                {
                    return false;
                }
            }
            else
            {
                throw new Exception("Printer/Device not instantiated");
            }
        }
        /// <summary>
        /// Basic Copy making with number of copy selections
        /// </summary>
        /// <TODO>add more copy parameters...</TODO>
        /// <param name="numCopies"></param>
        /// <returns></returns>
        public bool makeCopy(string numCopies)
        {
            if (device != null)
            {
                //sendCommand(Cartographer.MainDictionary["home"]);
                device.PowerManagement.Wake();

moveAndPress(deviceMap.JediMainDictionary800x600["CopyApp"]);
                device.ControlPanel.WaitForForm("CopyAppMainForm");
//while not on copy form
```

```
moveAndPress(deviceMap.JediCopyDictionary800x600["copies"]);
                device.ControlPanel.WaitForForm("HPNumericKeypad");

moveAndPress(deviceMap.JediHPNumericKeypadDictionary800x600["mButton" +
numCopies]);

moveAndPress(deviceMap.JediHPNumericKeypadDictionary800x600["mButtonOK"]);
                device.ControlPanel.WaitForForm("CopyAppMainForm");
//while not on copy form

moveAndPress(deviceMap.JediCopyDictionary800x600["startCopy"]);
                return true;
            }
            else
            {
                throw new Exception("Printer/Device not Instantiated");
            }
        }
        /// <summary>
        /// Types on Home page
        /// </summary>
        /// <param name="button"></param>
        /// <returns></returns>
        public bool pressHomepage(string button)
        {
            try
            {
                if
(deviceMap.JediMainDictionary800x600.ContainsKey(button))
                {
                    System.Drawing.Point ButtonLocation =
deviceMap.JediMainDictionary800x600[button];
                    //move and press myPoint
                    if (moveAndPress(ButtonLocation))
                    {
                        return true;
                    }
                    else
                    {
                        return false; //for whatever reason the move
reported unsuccessful
                    }
                }
                else
                {
                    Console.WriteLine("Could not find button.");
                    return false;
                }
            }
            catch (WebException)
            {
                return false;
            }
```

```
            catch (Exception e)
            {
                Console.WriteLine("Bummer: " + e.Message);
                Console.ReadLine();
                return false;
            }
        }


        /// <summary>
        /// Type on Virtual Keyboard, checks for casing and special
characters
        /// </summary>
        /// <param name="button"></param>
        /// <returns></returns>
        public bool pressKeyboard(string button)
        {
            if
(deviceMap.JediKeyboardDictionary800x600.ContainsKey(button))
            {//Consider alternative implementation using regular
expressions
                if (button.Length == 1) //logic for single character
buttons
                {// logic for shifted characters
                    if ((button[0] > 32 && button[0] < 44)
                        || button[0] == 58
                        || (button[0] > 59 && button[0] < 64)
                        || (button[0] > 64 && button[0] < 91)
                        || (button[0] > 93 && button[0] < 96)
                        || (button[0] > 122 && button[0] < 127))
                    {
                        shift();
                        return
moveAndPress(deviceMap.JediKeyboardDictionary800x600[button]);
                    }
                    else
                    {
                        return
moveAndPress(deviceMap.JediKeyboardDictionary800x600[button]); //not
shifted
                    }
                }
                else // logic for regular buttons (e.g., enter, tab, etc.)
                {
                    return
moveAndPress(deviceMap.JediKeyboardDictionary800x600[button]);
                }
            }
            else
            {
                return false;
            }
        }
```

```
        /// <summary>
        /// Press number on numpad
        /// </summary>
        /// <param name="number"></param>
        /// <returns></returns>
        private bool pressNumpad(string number)
        {
            if
(deviceMap.JediHPNumericKeypadDictionary800x600.ContainsKey(number))
            {

moveAndPress(deviceMap.JediHPNumericKeypadDictionary800x600[number]);
                return true;
            }
            else
            {
                throw new ArgumentException("Please only enter 0-9");
            }
        }

        /// <summary>
        /// hpacSecurePullPrint
        /// </summary>
        /// <param name="user"></param>
        /// <param name="password"></param>
        /// <returns></returns>
        public bool hpacPullPrintIRM(string auth)
        {
            if (device != null)
            {
                device.PowerManagement.Wake();
                if (pressHomepage("hpacsecurepullprint"))
                {
                    device.ControlPanel.WaitForForm("SignInForm");
                    enterAuthCode(auth);
                    pressKeyboard("mButtonOK");

device.ControlPanel.WaitForForm("OxpUIAppMainForm800X600");

moveAndPress(deviceMap.OxpUIAppMainFormDictionary800X600["firstDocument"])
;
                    return true;
                }
                else
                {
                    return false;
                }
            }
            else
            {
                throw new Exception("Printer/Device not instantiated.
Don't try to do operations with an uninstantiated device!");
            }
        }
```

```
        public bool hpacPullPrint(string user, string password)
        {
            if (device != null)
            {
                device.PowerManagement.Wake();
                pressHomepage("hpacsecurepullprint");

                device.ControlPanel.WaitForForm("SignInForm");

                enterUser(user);

                enterPassword(password);

                pressKeyboard("mButtonOK");


device.ControlPanel.WaitForForm("OxpUIAppMainForm800X600");


moveAndPress(deviceMap.OxpUIAppMainFormDictionary800X600["firstDocument"])
;

                return true;
            }
            else
            {
                throw new Exception("Printer/Device not instantiated.
Don't try to do operations with an uninstantiated device!");
            }
        }



        /// <summary>
        /// Enter Auth Code for HPAC using IRM
        /// </summary>
        /// <param name="code"></param>
        /// <returns></returns>
        private bool enterAuthCode(string code)
        {

moveAndPress(deviceMap.JediSignInFormDictionary800x600["authField"]);

            foreach (char number in code)
            {
                if (!pressKeyboard(number.ToString()))
                {
                    return false;
                }
            }
            return true;
        }
        /// <summary>
```

```
        /// Enter User for HPAC...X,Y coordinates transmitted over network
and not as User cleartext
        /// </summary>
        /// <param name="user"></param>
        /// <returns></returns>
        private bool enterUser(string user)
        {

moveAndPress(deviceMap.JediSignInFormDictionary800x600["userField"]);

            foreach (char letter in user)
            {
                if (!pressKeyboard(letter.ToString()))
                {
                    return false;
                }
            }
            return true;
        }

        /// <summary>
        /// Enter Password...X,Y coordinates transmitted over network and
not as password cleartext
        /// </summary>
        /// <param name="password"></param>
        /// <returns></returns>
        private bool enterPassword(string password)
        {

moveAndPress(deviceMap.JediSignInFormDictionary800x600["passwordField"]);

            foreach (char letter in password)
            {
                if (!pressKeyboard(letter.ToString()))
                {
                    return false;
                }
            }
            return true;
        }

        /// <summary>
        /// press shift key
        /// </summary>
        /// <returns></returns>
        private bool shift()
        {
            try
            {
                System.Drawing.Point myPoint =
deviceMap.JediKeyboardDictionary800x600["shift"];
                if (moveAndPress(myPoint))
                {
                    return true;
```

```
                }
                else
                {
                    return false;
                }
            }
            catch (WebException)
            {
                return false;
            }
        }
        /// <summary>
        /// Calibrates robot on startup
        /// </summary>
        /// <returns></returns>
        public bool initialize()
        {
            try
            {
                //calibrate robot
                string sURL;
                sURL = string.Format("http://{0}:{1}/api/Initialize",
robotIP, roboPort);
                WebRequest wrPOSTURL;
                wrPOSTURL = WebRequest.Create(sURL);
                wrPOSTURL.Credentials = credentialCache;
                wrPOSTURL.ContentType = "application/x-www-form-
urlencoded";
                wrPOSTURL.Method = "POST";
                wrPOSTURL.ContentLength = 0;
                wrPOSTURL.PreAuthenticate = true;
                //wrGETURL.Timeout = 3;
                Stream objStream;
                objStream = wrPOSTURL.GetResponse().GetResponseStream();
                objStream.Close();
                initialized = true;
                return true;
            }
            catch (WebException)
            {
                throw new WebException("An error occured communicating
with the robot, ensure server is on and correct modules exposed.");
            }
        }

        public bool pressXY800x600(System.Drawing.Point PressLocation)
        {
            if (PressLocation.X <= 800 && PressLocation.X >= 0 &&
PressLocation.Y <= 600 && PressLocation.Y >= 0)
            {
                moveAndPress(PressLocation);
                return true;
            }
            else
```

```
            {
                return false;
            }
        }

        /// <summary>
        /// Tells robot to move to xy and press
        /// </summary>
        /// <param name="PressLocation"></param>
        /// <returns> boolean </returns>
        private bool moveAndPress(System.Drawing.Point PressLocation)
        {
            try
            {
                if (!initialized)
                {
                    initialized = initialize();
                }
                string sURL;
                double[] coords = { 0, 0 };
                coords = deviceMap.convertCoordinates(PressLocation.X,
PressLocation.Y);
                Stream objStream;
                WebRequest wrPOSTURL;
                if (lastMove[0] != coords[0] || lastMove[1] != coords[1])
//if same location, skip move command
                {
                    lastMove[0] = coords[0];
                    lastMove[1] = coords[1];
                    //move to xy coordinates
                    sURL =
string.Format("http://{0}:{1}/api/MoveXY/{2}/{3}", robotIP, roboPort,
coords[0].ToString(), coords[1].ToString());
                    wrPOSTURL = WebRequest.Create(sURL);
                    wrPOSTURL.PreAuthenticate = true;
                    wrPOSTURL.Credentials = credentialCache;
                    wrPOSTURL.ContentType = "application/x-www-form-
urlencoded";
                    wrPOSTURL.Method = "POST";
                    wrPOSTURL.ContentLength = 0;
                    objStream =
wrPOSTURL.GetResponse().GetResponseStream();
                    objStream.Close();
                }
                //press button
                sURL = string.Format("http://{0}:{1}/api/Press/{2}/{3}",
robotIP, roboPort, "5", "0.2");
                wrPOSTURL = WebRequest.Create(sURL);
                wrPOSTURL.PreAuthenticate = true;
                wrPOSTURL.Credentials = credentialCache;
                wrPOSTURL.ContentType = "application/x-www-form-
urlencoded";
                wrPOSTURL.Method = "POST";
                wrPOSTURL.ContentLength = 0;
```

```
                objStream = wrPOSTURL.GetResponse().GetResponseStream();
                objStream.Close();
                return true;
            }
            catch (WebException)    //can be thrown by the webrequests
            {
                Console.WriteLine("An error occured communicating with the
robot, ensure server is on and correct modules exposed.");
                throw new WebException("An error occured communicating
with the robot, ensure server is on and correct modules exposed.");
            }
            catch (ArgumentException) //can be thrown by
Cartographer.convertCoordinates
            {
                Console.WriteLine("An error occured converting the
coordinates. Ensure validity of original coordinates");
                throw new ArgumentException("An error occured converting
the coordinates. Ensure validity of original coordinates");
            }
            catch (Exception)
            {
                Console.WriteLine("An error occured communicating with the
robot, ensure server is on and correct modules exposed. Additionally,
ensure requested coordinates are within bounds.");
                throw new Exception("An error occured communicating with
the robot, ensure server is on and correct modules exposed. Additionally,
ensure requested coordinates are within bounds.");
            }
        }
        public string getProperty(string id, string property)
        {
            return device.ControlPanel.GetProperty(id, property);
        }
        public int getOffsetIterativeParallel(string controlName)
        {
            List<string> controls = new List<string>();
            int offset = 0;
            string parent = getProperty(controlName, "Parent");

            while (parent != "[null]")
            {
                controls.Add(controlName);
                parent = getProperty(controlName, "Parent");
                if (parent != "[null]")
                    controlName = getProperty(controlName, "Parent.Name");
            }
            Parallel.ForEach(controls,
                (control) =>
                {
                    try
                    {
                        //Entry Section
                        int localOffset =
Convert.ToInt32(getProperty(control, "Top"));
```

```
                        //Critical Section
                        propertiesLock.EnterWriteLock();
                        offset += localOffset;
                    }
                    finally
                    {
                        //function absolutely must release lock regardless
of successful completion
                        propertiesLock.ExitWriteLock();
                        //Remainder Section
                    }
                });
            return offset;
        }
        public int getOffsetRecursive(string controlName)
        {
            string parent = getProperty(controlName, "Parent");
            if (parent != "[null]")
            {
                int offset = getOffsetRecursive(getProperty(controlName,
"Parent.Name"));
                int myHeight = Convert.ToInt32(getProperty(controlName,
"Top"));
                return offset + myHeight;
            }
            else //basecase
            {
                int parentHeight =
Convert.ToInt32(getProperty(controlName, "Top"));
                return parentHeight;
            }
        }
    }
}
```

Cartographer.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace RobotLibrary
{
    public class Cartographer
    {
        public readonly Dictionary<string, System.Drawing.Point>
JediSignInFormDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
        {
            { "userField", new System.Drawing.Point(100, 170)},
            { "passwordField", new System.Drawing.Point(100, 255) },
            { "authField", new System.Drawing.Point(100,130) }
        };
```

```
        public readonly Dictionary<string, System.Drawing.Point>
JediHPMessageBoxDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
        {
            //needs updated!
            { "ok", new System.Drawing.Point(660, 440)},
            { "activeJobs", new System.Drawing.Point(500, 440) }
        };
        public readonly Dictionary<string, System.Drawing.Point>
JediBaseJobStartPopupDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
        {
            { "joblog", new System.Drawing.Point(275, 440)},
            { "ok", new System.Drawing.Point(470, 440)},
            { "cancel", new System.Drawing.Point(640, 440) }
        };
        public readonly Dictionary<string, System.Drawing.Point>
JediTwoButtonMessageBoxDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
        {
            { "ok", new System.Drawing.Point(510, 440)},
            { "cancel", new System.Drawing.Point(660, 440)}
        };
        public readonly Dictionary<string, System.Drawing.Point>
JediScanToDeviceDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
        {
            { "pause", new System.Drawing.Point(160, 30)},
            { "startscan", new System.Drawing.Point(225, 25)},
            { "returnHome", new System.Drawing.Point(40, 25)},
            { "firstfolder", new System.Drawing.Point(115, 230)},
            { "jobname", new System.Drawing.Point(435, 215)}
        };
        public readonly Dictionary<string, System.Drawing.Point>
JediRetrieveFromDeviceDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
        {
            { "pause", new System.Drawing.Point(160, 30)},
            { "startretrieve", new System.Drawing.Point(225, 25)},
            { "returnHome", new System.Drawing.Point(40, 25)},
            { "firstfolder", new System.Drawing.Point(115, 225)},
            { "jobname", new System.Drawing.Point(435, 215)},
            { "all", new System.Drawing.Point(165, 225) },
            {  "firstDoc", new System.Drawing.Point(175, 285)},
            { "ok", new System.Drawing.Point(660, 440)},
            { "delete", new System.Drawing.Point(105, 565)}
        };
        public readonly Dictionary<string, System.Drawing.Point>
JediCopyDictionary800x600 = new Dictionary<string, System.Drawing.Point>()
        {
            { "startCopy", new System.Drawing.Point(160,20)},
            { "copies", new System.Drawing.Point(755,85)},
            { "gohome", new System.Drawing.Point(35, 35)}
        };
```

```
        public readonly Dictionary<string, System.Drawing.Point>
OxpUIAppMainFormDictionary800X600 = new Dictionary<string,
System.Drawing.Point>()
        {
            { "refresh", new System.Drawing.Point(715,405)},
            { "home", new System.Drawing.Point(32,28 )},
            { "firstDocument", new System.Drawing.Point(40, 270)}
        };
        public readonly Dictionary<string, System.Drawing.Point>
JediPullPrintLocalDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
        {
             { "firstDocument", new System.Drawing.Point(40,180)},
             { "secondDocument", new System.Drawing.Point(40,225 )},
             { "thirdDocument", new System.Drawing.Point(40, 270)},
             { "fourthDocument", new System.Drawing.Point(40, 315)},
             { "returnHome", new System.Drawing.Point(40, 25)},
             { "printFiles", new System.Drawing.Point(700, 570)},
             { "pullPrintDemo", new System.Drawing.Point(160, 30)}
         };
        public readonly Dictionary<string, System.Drawing.Point>
JediMainDictionary800x600 = new Dictionary<string, System.Drawing.Point>()
            {
                { "mResetButton", new System.Drawing.Point(22,85)},
                { "mSignInButton", new System.Drawing.Point(145, 85)},
                { "m_TitleHelpButton", new System.Drawing.Point(770, 30)},
                { "NumberOfCopies", new System.Drawing.Point(750, 85)},
                { "hpacsecurepullprint", new System.Drawing.Point(430,
200)},
                { "CopyApp", new System.Drawing.Point(60, 350)},
                { "pullprintlocal", new System.Drawing.Point(60, 200)},
                { "SendFaxApp", new System.Drawing.Point(430, 350)},
                { "SaveToDeviceMemoryApp", new System.Drawing.Point(60,
490)},
                { "OpenFromDeviceMemoryApp", new System.Drawing.Point(430,
490)},
                { "mLanguageSelectButton", new System.Drawing.Point(620,
30)},
                { "mSleepButton", new System.Drawing.Point(660, 30)},
                { "mDeviceInformationButton", new
System.Drawing.Point(710, 30)},
                { "mStopButton", new System.Drawing.Point(100, 30)},
                { "mStartButton", new System.Drawing.Point(160, 30)},
                //{ "mResetButton", new System.Drawing.Point(770, 570)},
                { "home", new System.Drawing.Point(0, 0)},
                { "far", new System.Drawing.Point(800, 600)}
            };
        public readonly Dictionary<string, System.Drawing.Point>
JediHPNumericKeypadDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
            {
                { "mButton1", new System.Drawing.Point(270,240)},
                { "mButton2", new System.Drawing.Point(320, 240)},
                { "mButton3", new System.Drawing.Point(365, 240)},
```

```
                { "mButton4", new System.Drawing.Point(270, 290)},
                { "mButton5", new System.Drawing.Point(320, 290)},
                { "mButton6", new System.Drawing.Point(365, 290)},
                { "mButton7", new System.Drawing.Point(270, 340)},
                { "mButton8", new System.Drawing.Point(320, 340)},
                { "mButton9", new System.Drawing.Point(365, 340)},
                { "mButton0", new System.Drawing.Point(320, 390)},
                { "mButtonOK", new System.Drawing.Point(325, 445)},
                { "mButtonCancel", new System.Drawing.Point(480, 445)},
                { "mButtonLeft", new System.Drawing.Point(486, 340)},
                { "mButtonRight", new System.Drawing.Point(530, 340)},
                { "mButtonBackspace", new System.Drawing.Point(510, 240)},
                { "mButtonClear", new System.Drawing.Point(530, 290)}
            };
        public readonly Dictionary<string, System.Drawing.Point>
JediKeyboardDictionary800x600 = new Dictionary<string,
System.Drawing.Point>()
            {
            //default keyboard
            { "`", new System.Drawing.Point(30, 400)},
            { "1", new System.Drawing.Point(85, 400)},
            { "2", new System.Drawing.Point(140, 400)},
            { "3", new System.Drawing.Point(195, 400)},
            { "4", new System.Drawing.Point(250, 400)},
            { "5", new System.Drawing.Point(305, 400)},
            { "6", new System.Drawing.Point(360, 400)},
            { "7", new System.Drawing.Point(415, 400)},
            { "8", new System.Drawing.Point(470, 400)},
            { "9", new System.Drawing.Point(525, 400)},
            { "0", new System.Drawing.Point(580, 400)},
            { "-", new System.Drawing.Point(635, 400)},
            { "=", new System.Drawing.Point(690, 400)},
            { "backspace", new System.Drawing.Point(755, 400)},

            //shifted keyboard
            { "~", new System.Drawing.Point(30, 400)},
            { "!", new System.Drawing.Point(85, 400)},
            //@ is represented discretely elsewhere on the keyboard
            { "#", new System.Drawing.Point(195, 400)},
            { "$", new System.Drawing.Point(250, 400)},
            { "%", new System.Drawing.Point(305, 400)},
            { "^", new System.Drawing.Point(360, 400)},
            { "&", new System.Drawing.Point(415, 400)},
            { "*", new System.Drawing.Point(470, 400)},
            { "(", new System.Drawing.Point(525, 400)},
            { ")", new System.Drawing.Point(580, 400)},
            //"_" is represented with its own key elsewhere
            { "+", new System.Drawing.Point(690, 400)},

            //default keyboard
            { "a", new System.Drawing.Point(130, 490)},
            { "s", new System.Drawing.Point(185, 490)},
            { "d", new System.Drawing.Point(240, 490)},
            { "f", new System.Drawing.Point(295, 490)},
```

```
{ "g", new System.Drawing.Point(350, 490)},
{ "h", new System.Drawing.Point(405, 490)},
{ "j", new System.Drawing.Point(460, 490)},
{ "k", new System.Drawing.Point(515, 490)},
{ "l", new System.Drawing.Point(580, 490)},
{ ";", new System.Drawing.Point(635, 490)},
{ "'", new System.Drawing.Point(690, 490)},

//shifted keyboard
{ "A", new System.Drawing.Point(130, 490)},
{ "S", new System.Drawing.Point(185, 490)},
{ "D", new System.Drawing.Point(240, 490)},
{ "F", new System.Drawing.Point(295, 490)},
{ "G", new System.Drawing.Point(350, 490)},
{ "H", new System.Drawing.Point(405, 490)},
{ "J", new System.Drawing.Point(460, 490)},
{ "K", new System.Drawing.Point(515, 490)},
{ "L", new System.Drawing.Point(580, 490)},
{ ":", new System.Drawing.Point(635, 490)},
{ "\"", new System.Drawing.Point(690, 490)},

//default keyboard
{ "z", new System.Drawing.Point(170, 530)},
{ "x", new System.Drawing.Point(225, 530)},
{ "c", new System.Drawing.Point(280, 530)},
{ "v", new System.Drawing.Point(335, 530)},
{ "b", new System.Drawing.Point(390, 530)},
{ "n", new System.Drawing.Point(445, 530)},
{ "m", new System.Drawing.Point(500, 530)},
{ ",", new System.Drawing.Point(555, 530)},
{ ".", new System.Drawing.Point(610, 530)},
{ "/", new System.Drawing.Point(675, 530)},

//shifted keyboard
{ "Z", new System.Drawing.Point(170, 530)},
{ "X", new System.Drawing.Point(225, 530)},
{ "C", new System.Drawing.Point(280, 530)},
{ "V", new System.Drawing.Point(335, 530)},
{ "B", new System.Drawing.Point(390, 530)},
{ "N", new System.Drawing.Point(445, 530)},
{ "M", new System.Drawing.Point(500, 530)},
{ "<", new System.Drawing.Point(555, 530)},
{ ">", new System.Drawing.Point(610, 530)},
{ "?", new System.Drawing.Point(675, 530)},

//default keyboard
{ "language", new System.Drawing.Point(50, 440)},
{ "q", new System.Drawing.Point(115, 440)},
{ "w", new System.Drawing.Point(170, 440)},
{ "e", new System.Drawing.Point(225, 440)},
{ "r", new System.Drawing.Point(280, 440)},
{ "t", new System.Drawing.Point(335, 440)},
{ "y", new System.Drawing.Point(390, 440)},
{ "u", new System.Drawing.Point(445, 440)},
```

```
            { "i", new System.Drawing.Point(500, 440)},
            { "o", new System.Drawing.Point(555, 440)},
            { "p", new System.Drawing.Point(610, 440)},
            { "[", new System.Drawing.Point(665, 440)},
            { "]", new System.Drawing.Point(720, 440)},
            { "\\", new System.Drawing.Point(775, 440)},

            //shifted keyboard
            { "Q", new System.Drawing.Point(115, 440)},
            { "W", new System.Drawing.Point(170, 440)},
            { "E", new System.Drawing.Point(225, 440)},
            { "R", new System.Drawing.Point(280, 440)},
            { "T", new System.Drawing.Point(335, 440)},
            { "Y", new System.Drawing.Point(390, 440)},
            { "U", new System.Drawing.Point(445, 440)},
            { "I", new System.Drawing.Point(500, 440)},
            { "O", new System.Drawing.Point(555, 440)},
            { "P", new System.Drawing.Point(610, 440)},
            { "{", new System.Drawing.Point(665, 440)},
            { "}", new System.Drawing.Point(720, 440)},
            { "|", new System.Drawing.Point(775, 440)},

            //both keyboards
            { "mButtonOK", new System.Drawing.Point(515, 575)},
            { "cancel", new System.Drawing.Point(670, 575)},
            { "caps", new System.Drawing.Point(50, 480)},
            { "alt", new System.Drawing.Point(45, 570)},
            { "123", new System.Drawing.Point(750, 530)},
            { "@", new System.Drawing.Point(110, 575)},
            { "shift", new System.Drawing.Point(40, 530)},
            { "_", new System.Drawing.Point(220, 575)},
            { "userField", new System.Drawing.Point(75, 170)},
            { "passwordField", new System.Drawing.Point(75, 255)},
            { "erase", new System.Drawing.Point(770, 575)},
            { "enter", new System.Drawing.Point(750, 485)}
        };

        private int maxBoxExtentX;
        private int maxBoxExtentY;
        private int minBoxExtentX = 0;
        private int minBoxExtentY = 0;
        private double robotMaxExtentX;
        private double robotMaxExtentY;
        //always greater than or equal to zero...the default is zero
        //sometimes you want want to shrink the coordinate system to fit
the device screen
        private double robotMinExtentX;
        private double robotMinExtentY;
        private double optimizedX; //= ((robotMaxExtentX -
robotMinExtentX) / (maxBoxExtentX - minBoxExtentX));
        private double optimizedY; //= ((robotMaxExtentY -
robotMinExtentY) / (maxBoxExtentY - minBoxExtentY));
        private readonly Stopwatch stopwatch = new Stopwatch();
```

```
        public Cartographer(int maxBoxExtentX = 800, int maxBoxExtentY =
600, int minBoxExtentX = 0, int minBoxExtentY = 0, double robotMaxExtentX
= 170, double robotMaxExtentY = 141.0, double robotMinExtentX = 10, double
robotMinExtentY = 22)
        {
            this.maxBoxExtentX = maxBoxExtentX;
            this.maxBoxExtentY = maxBoxExtentY;
            this.minBoxExtentX = minBoxExtentX;
            this.minBoxExtentY = minBoxExtentY;
            this.robotMaxExtentX = robotMaxExtentX;
            this.robotMaxExtentY = robotMaxExtentY;
            this.robotMinExtentX = robotMinExtentX;
            this.robotMinExtentY = robotMinExtentY;
            this.optimizedX = ((robotMaxExtentX - robotMinExtentX) /
(maxBoxExtentX - minBoxExtentX));
            this.optimizedY = ((robotMaxExtentY - robotMinExtentY) /
(maxBoxExtentY - minBoxExtentY));


        }
        /* //deprecated due to high startup cost of parallelization far
outweighing the benefit of the parallel operations
         * //experimentally discovered an average of 300 ticks in serial
and an average of 10000+ ticks in parallel
        public double[] convertCoordinatesParallel(double x, double y)
        {
            double[] coords = { 0, 0 };
            //inverted min-max normalization
            //Standard min-max normalization -> v' = (v - min)/(max-
min)(new_max-new_min) + new_min
            //To invert, subtract the old maximum instead of the old
minimum from the numerator and take absolute value
            //Inverted min-max normalization -> v' = (v - max)etc...
            if (((maxBoxExtentX - minBoxExtentX) == 0) || ((maxBoxExtentY
- minBoxExtentY) == 0))
            {
                throw new ArithmeticException("Your max extent and min
extent are the same.\nThe area of a line or point is always zero.\nYour
coordinate system should not be a line or a single point.\nPlease specify
valid extents.");
            }
            if (x <= maxBoxExtentX && x >= minBoxExtentX && y >=
minBoxExtentY && y <= maxBoxExtentY)
            {
                stopwatch.Restart();
                Parallel.Invoke(
                    () => coords[0] = Math.Abs(Math.Round((x -
maxBoxExtentX) * optimizedX - robotMinExtentX)),
                    () => coords[1] = Math.Abs(Math.Round((y -
maxBoxExtentY) * optimizedY - robotMinExtentY)));
                stopwatch.Stop();
                TimeSpan ts = stopwatch.Elapsed;
```

```
                Console.WriteLine(string.Format("Converted ({0},{1},) to
({2},{3}) in {4} ticks or {5} ms", x, y, coords[0], coords[1], ts.Ticks,
ts.TotalMilliseconds));
                return coords;
            }
            else
            {
                Console.WriteLine("Your coordinates are outside the
accepted range");
                return coords;
            }
        }
        */
        public double[] convertCoordinates(double x, double y)
        {
            double[] coords = { 0, 0 };
            //inverted min-max normalization
            //Standard min-max normalization -> v' = (v - min)/(max-
min)(new_max-new_min) + new_min
            //To invert, subtract the old maximum instead of the old
minimum from the numerator and take absolute value
            //Inverted min-max normalization -> v' = (v - max)etc...
            if (((maxBoxExtentX - minBoxExtentX) == 0) || ((maxBoxExtentY
- minBoxExtentY) == 0))
            {
                throw new ArithmeticException("Your max extent and min
extent are the same.\nThe area of a line or point is always zero.\nYour
coordinate system should not be a line or a single point.\nPlease specify
valid extents.");
            }
            if (x <= maxBoxExtentX && x >= minBoxExtentX && y >=
minBoxExtentY && y <= maxBoxExtentY)
            {
                stopwatch.Restart();

                coords[0] = Math.Abs(Math.Round((x - maxBoxExtentX) *
optimizedX - robotMinExtentX));
                coords[1] = Math.Abs(Math.Round((y - maxBoxExtentY) *
optimizedY - robotMinExtentY));
                stopwatch.Stop();
                TimeSpan ts = stopwatch.Elapsed;
                Console.WriteLine(string.Format("Converted ({0},{1},) to
({2},{3}) in {4} ticks or {5} ms", x, y, coords[0], coords[1], ts.Ticks,
ts.TotalMilliseconds));
                return coords;
            }
            else
            {
                Console.WriteLine("Your coordinates are outside the
accepted range");
                return coords;
            }
        }
    }
```

```
}
```

Appendix C: Robot Demo

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace RobotDemo
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            try
            {
                Application.EnableVisualStyles();
                Application.SetCompatibleTextRenderingDefault(false);
                Application.Run(new RobotDemo());
            }
            catch (Exception)
                { }
        }
    }
}
```

RobotDemo.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HP.DeviceAutomation.Jedi;
using HP.ScalableTest.Print;
using RobotLibrary;

namespace RobotDemo
{
    public partial class RobotDemo : Form
```

```
    {
        private JediWindjammerDevice device;
        private JediWindjammerRobot robot;
        private bool connected = false;
        private bool calibrated = false;
        private short countdown = 29;

        public int getControlTrueX(Control control)
        {
            if(control.Parent != null)
            {
                return control.Location.X +
getControlTrueX(control.Parent);
            }
            else
            {
                return control.Location.X;
            }

        }

        public RobotDemo()
        {
            try
            {
                InitializeComponent();
                ClockBox.Enabled = false;
                ScreenTimer.Interval = 1000;
                ScreenTimer.Tick += screenRefreshTimer_Tick;
                DemoTimer.Interval = 30000;
                DemoTimer.Tick += startDemo_Tick;
                UpdateClockTimer.Interval = 985;
                UpdateClockTimer.Tick += updateClock;
                int location = getControlTrueX(StopDemoButton);
                location = location + 0;
            }
            catch (Exception)
            {

            }
        }


        private  void updateClock(object sender, EventArgs e)
        {
            if (countdown < 0)
            {
                UpdateClockTimer.Stop();
            }
            else {

                ClockBox.Text = countdown.ToString();
                countdown--;
```

```
            }
        }
        private async void startDemo_Tick(object sender, EventArgs e)
        {
            if (connected && robot != null)
            {
                try
                {
                    StartDemoButton.Enabled = false;
                    StopDemoButton.Enabled = false;
                    UpdateClockTimer.Stop();
                    DemoTimer.Stop();
                    await Task.Run(() =>
                    {
                        robot.scanToDeviceMemory("demofile");
                        robot.retrieveFromDeviceMemory("firstDoc", true);
                    });
                    DemoTimer.Start();
                    countdown = 30;
                    ClockBox.Text = countdown.ToString();
                    countdown--;
                    UpdateClockTimer.Start();
                    StartDemoButton.Enabled = true;
                    StopDemoButton.Enabled = true;
                }
                catch (WebException we)
                {
                    DemoTimer.Stop();
                    UpdateClockTimer.Stop();
                    string title = "Lost Connectivity";
                    MessageBoxButtons warning = MessageBoxButtons.OK;
                    DialogResult result;
                    result = MessageBox.Show(ConnectButton, we.Message,
title, warning);
                    connected = false;
                    ConnectButton.Text = "Retry";
                    ConnectButton.Enabled = true;
                }
            }
        }
        /*
        private void SetBySocket(IPAddress address)
        {
            using (Socket ipSocket = new Socket(address.AddressFamily,
SocketType.Stream, ProtocolType.Tcp))
            {
                ipSocket.Connect(address, 9100);

                byte[] msg =
System.Text.Encoding.UTF8.GetBytes(Resource.PJL_PAPERLESS_MODE_ON_JEDI);
                ipSocket.Send(msg);
                ipSocket.Close();
            }
        }
```

```
        */
        private async void ConnectButton_Click(object sender, EventArgs e)
        {
            try
            {
                if (!connected)
                {
                    ConnectButton.Enabled = false; //disable button to
prevent reentrancy
                    ConnectButton.Text = "Connecting";
                    await Task.Run(async () =>
                    {
                        device = new
JediWindjammerDevice(DeviceIpCheckBox.Text, "NotARealPassword");
                        robot = new JediWindjammerRobot(device,
RobotIpTextBox.Text);
                        if (device != null && robot != null)
                        {
                            Task paperlessPrinting =
ConfigurePrinterAsync();
                            Task<Image> getDeviceImage =
UpdateDeviceScreenAsync();
                            Task initializeRobot = InitializeRobotAsync();

                            DeviceScreen.Image = await getDeviceImage;
                            await initializeRobot;
                            await paperlessPrinting;
                        }
                        else
                        {
                            throw new Exception();
                        }

                    });
                    RobotIpTextBox.Enabled = false;
                    DeviceIpCheckBox.Enabled = false;
                    ConnectButton.Text = "Connected";
                    StartDemoButton.Enabled = true;
                    StopDemoButton.Enabled = true;
                    RecalibrateButton.Enabled = true;
                    ScreenTimer.Start();
                    connected = true;
                    calibrated = true;
                    ConnectButton.Enabled = false;
                }
            }
            catch (Exception)
            {
                string message = "An error occured instantiating the
device and/or robot. Please check the ip address and network connection";
                string title = "Robot or Device Not Ready";
                MessageBoxButtons warning = MessageBoxButtons.OK;
                DialogResult result;
```

```
                result = MessageBox.Show(ConnectButton, message , title,
warning);
                connected = false;
                ConnectButton.Text = "Retry";
                ConnectButton.Enabled = true;
            }
        }
        async Task ConfigurePrinterAsync()
        {
            await Task.Run(() =>
PrintDeviceConfiguration.SetPaperlessPrintMode(DeviceIpCheckBox.Text,
device.AdminPassword, 9100, true));
        }

        async Task InitializeRobotAsync()
        {
            calibrated = false;
            await Task.Run(() => {  robot.initialize(); });
            calibrated = true;
        }

        async Task<Image> UpdateDeviceScreenAsync()
        {

            Image myImage = null;
                await Task.Run(() => { device.PowerManagement.Wake();
myImage = device.ControlPanel.ScreenCapture(); });
                return myImage;

        }
        private async void screenRefreshTimer_Tick(object sender,
EventArgs e)
        {
            ScreenTimer.Enabled = false;
            if (device != null)
            {
                await Task.Run(() => device.PowerManagement.Wake());
                DeviceScreen.Image = await Task.Run(() =>
device.ControlPanel.ScreenCapture());
            }
            ScreenTimer.Enabled = true;
        }

        private void DeviceScreen_MouseDoubleClick(object sender,
MouseEventArgs e)
        {
            if (calibrated)
            {
                try
                {
                    robot.pressXY800x600(new System.Drawing.Point(e.X,
e.Y));
                }
                catch (WebException we)
```

```
                {
                    string title = "Could not communicate with robot";
                    MessageBoxButtons warning = MessageBoxButtons.OK;
                    DialogResult result;
                    result = MessageBox.Show(this, we.Message, title,
warning);
                    connected = false;
                    ConnectButton.Text = "Retry";
                    ConnectButton.Enabled = true;
                }
                catch (ArgumentException ae)
                {
                    string title = "Requested Coordinates Out of Bounds";
                    MessageBoxButtons warning = MessageBoxButtons.OK;
                    DialogResult result;
                    result = MessageBox.Show(this, ae.Message, title,
warning);
                    connected = false;
                    ConnectButton.Text = "Retry";
                    ConnectButton.Enabled = true;
                }
            }
        }

        private void StartDemoButton_Click(object sender, EventArgs e)
        {
            DemoTimer.Start();
            countdown = 30;
            ClockBox.Text = countdown.ToString();
            countdown--;
            UpdateClockTimer.Start();
        }

        private void StopDemoButton_Click(object sender, EventArgs e)
        {
            DemoTimer.Stop();
            UpdateClockTimer.Stop();
        }

        private void DeviceScreen_MouseMove(object sender, MouseEventArgs
e)
        {
            if (connected && robot != null)
            {
                double[] coords = { 0, 0 };
                string mouseLocation = "Printer Device Screen";
                coords = robot.deviceMap.convertCoordinates(e.X, e.Y);
                mouseLocation = string.Format("({0},{1})", coords[0],
coords[1]);
                MouseLocation.Text = mouseLocation;
            }
        }
```

```
        private async void RecalibrateButton_Click(object sender,
EventArgs e)
        {
            RecalibrateButton.Enabled = false;
            DemoTimer.Stop();
            UpdateClockTimer.Stop();
            if (connected && robot != null)
            {
                try
                {
                    await InitializeRobotAsync();
                }
                catch (WebException we)
                {
                    string title = "Could not communicate with robot";
                    MessageBoxButtons warning = MessageBoxButtons.OK;
                    DialogResult result;
                    result = MessageBox.Show(this, we.Message, title,
warning);
                    connected = false;
                    ConnectButton.Enabled = true;
                }
            }
            RecalibrateButton.Enabled = true;
        }
    }
}
```

RobotDemo.Designer.cs

```
namespace RobotDemo
{
    partial class RobotDemo
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(RobotDemo));
            this.DeviceScreen = new System.Windows.Forms.PictureBox();
            this.RobotIpTextBox = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.RobotLabel = new System.Windows.Forms.Label();
            this.DeviceIpCheckBox = new System.Windows.Forms.TextBox();
            this.PrinterLabel = new System.Windows.Forms.Label();
            this.ConnectButton = new System.Windows.Forms.Button();
            this.ScreenTimer = new
System.Windows.Forms.Timer(this.components);
            this.StartDemoButton = new System.Windows.Forms.Button();
            this.StopDemoButton = new System.Windows.Forms.Button();
            this.DemoTimer = new
System.Windows.Forms.Timer(this.components);
            this.DeviceScreenBox = new System.Windows.Forms.GroupBox();
            this.MouseLocation = new System.Windows.Forms.Label();
            this.RecalibrateButton = new System.Windows.Forms.Button();
            this.UpdateClockTimer = new
System.Windows.Forms.Timer(this.components);
            this.ClockBox = new System.Windows.Forms.TextBox();

((System.ComponentModel.ISupportInitialize)(this.DeviceScreen)).BeginInit(
);
            this.DeviceScreenBox.SuspendLayout();
            this.SuspendLayout();
            //
            // DeviceScreen
            //
            this.DeviceScreen.BackColor =
System.Drawing.SystemColors.ActiveBorder;
            this.DeviceScreen.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("DeviceScreen.BackgroundImage"
)));
            this.DeviceScreen.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Center;
            this.DeviceScreen.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.DeviceScreen.Cursor = System.Windows.Forms.Cursors.Cross;
            this.DeviceScreen.InitialImage = null;
            this.DeviceScreen.Location = new System.Drawing.Point(1, 19);
            this.DeviceScreen.MaximumSize = new System.Drawing.Size(800,
600);
```

```
            this.DeviceScreen.MinimumSize = new System.Drawing.Size(800,
600);
            this.DeviceScreen.Name = "DeviceScreen";
            this.DeviceScreen.Size = new System.Drawing.Size(800, 600);
            this.DeviceScreen.TabIndex = 0;
            this.DeviceScreen.TabStop = false;
            this.DeviceScreen.MouseDoubleClick += new
System.Windows.Forms.MouseEventHandler(this.DeviceScreen_MouseDoubleClick)
;
            this.DeviceScreen.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.DeviceScreen_MouseMove);
            //
            // RobotIpTextBox
            //
            this.RobotIpTextBox.Location = new System.Drawing.Point(79,
39);
            this.RobotIpTextBox.Name = "RobotIpTextBox";
            this.RobotIpTextBox.Size = new System.Drawing.Size(100, 20);
            this.RobotIpTextBox.TabIndex = 1;
            this.RobotIpTextBox.Text = "192.168.1.2";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(76, 23);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(74, 16);
            this.label1.TabIndex = 3;
            this.label1.Text = "IP Address";
            //
            // RobotLabel
            //
            this.RobotLabel.AutoSize = true;
            this.RobotLabel.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.RobotLabel.Location = new System.Drawing.Point(8, 37);
            this.RobotLabel.Name = "RobotLabel";
            this.RobotLabel.Size = new System.Drawing.Size(53, 20);
            this.RobotLabel.TabIndex = 5;
            this.RobotLabel.Text = "Robot";
            //
            // DeviceIpCheckBox
            //
            this.DeviceIpCheckBox.Location = new System.Drawing.Point(79,
76);
            this.DeviceIpCheckBox.Name = "DeviceIpCheckBox";
            this.DeviceIpCheckBox.Size = new System.Drawing.Size(100, 20);
            this.DeviceIpCheckBox.TabIndex = 6;
            this.DeviceIpCheckBox.Text = "169.254.81.110";
            //
```

```
            // PrinterLabel
            //
            this.PrinterLabel.AutoSize = true;
            this.PrinterLabel.Font = new System.Drawing.Font("Microsoft
Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.PrinterLabel.Location = new System.Drawing.Point(8, 74);
            this.PrinterLabel.Name = "PrinterLabel";
            this.PrinterLabel.Size = new System.Drawing.Size(55, 20);
            this.PrinterLabel.TabIndex = 10;
            this.PrinterLabel.Text = "Printer";
            //
            // ConnectButton
            //
            this.ConnectButton.Font = new System.Drawing.Font("Microsoft
Sans Serif", 14.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.ConnectButton.Location = new System.Drawing.Point(199,
39);
            this.ConnectButton.Name = "ConnectButton";
            this.ConnectButton.Size = new System.Drawing.Size(118, 57);
            this.ConnectButton.TabIndex = 11;
            this.ConnectButton.Text = "Connect";
            this.ConnectButton.UseVisualStyleBackColor = true;
            this.ConnectButton.Click += new
System.EventHandler(this.ConnectButton_Click);
            //
            // ScreenTimer
            //
            this.ScreenTimer.Interval = 250;
            //
            // StartDemoButton
            //
            this.StartDemoButton.Enabled = false;
            this.StartDemoButton.Font = new System.Drawing.Font("Microsoft
Sans Serif", 14.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.StartDemoButton.Location = new System.Drawing.Point(541,
23);
            this.StartDemoButton.Name = "StartDemoButton";
            this.StartDemoButton.Size = new System.Drawing.Size(169, 39);
            this.StartDemoButton.TabIndex = 12;
            this.StartDemoButton.Text = "Start Demo Timer";
            this.StartDemoButton.UseVisualStyleBackColor = true;
            this.StartDemoButton.Click += new
System.EventHandler(this.StartDemoButton_Click);
            //
            // StopDemoButton
            //
            this.StopDemoButton.Enabled = false;
            this.StopDemoButton.Font = new System.Drawing.Font("Microsoft
Sans Serif", 14.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.StopDemoButton.Location = new System.Drawing.Point(541,
68);
            this.StopDemoButton.Name = "StopDemoButton";
            this.StopDemoButton.Size = new System.Drawing.Size(169, 40);
            this.StopDemoButton.TabIndex = 13;
            this.StopDemoButton.Text = "Stop Demo Timer";
            this.StopDemoButton.UseVisualStyleBackColor = true;
            this.StopDemoButton.Click += new
System.EventHandler(this.StopDemoButton_Click);
            //
            // DemoTimer
            //
            this.DemoTimer.Interval = 20000;
            //
            // DeviceScreenBox
            //
            this.DeviceScreenBox.Controls.Add(this.DeviceScreen);
            this.DeviceScreenBox.Controls.Add(this.MouseLocation);
            this.DeviceScreenBox.Dock =
System.Windows.Forms.DockStyle.Bottom;
            this.DeviceScreenBox.Font = new System.Drawing.Font("Microsoft
Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.DeviceScreenBox.Location = new System.Drawing.Point(0,
110);
            this.DeviceScreenBox.MaximumSize = new
System.Drawing.Size(802, 622);
            this.DeviceScreenBox.MinimumSize = new
System.Drawing.Size(802, 622);
            this.DeviceScreenBox.Name = "DeviceScreenBox";
            this.DeviceScreenBox.Size = new System.Drawing.Size(802, 622);
            this.DeviceScreenBox.TabIndex = 14;
            this.DeviceScreenBox.TabStop = false;
            this.DeviceScreenBox.Text = "Printer Device Screen";
            //
            // MouseLocation
            //
            this.MouseLocation.AutoSize = true;
            this.MouseLocation.Location = new System.Drawing.Point(165,
0);
            this.MouseLocation.Name = "MouseLocation";
            this.MouseLocation.Size = new System.Drawing.Size(41, 20);
            this.MouseLocation.TabIndex = 1;
            this.MouseLocation.Text = "(0,0)";
            this.MouseLocation.TextAlign =
System.Drawing.ContentAlignment.MiddleLeft;
            //
            // RecalibrateButton
            //
            this.RecalibrateButton.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Zoom;
            this.RecalibrateButton.Enabled = false;
            this.RecalibrateButton.Font = new
System.Drawing.Font("Microsoft Sans Serif", 14.25F,
```

```
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.RecalibrateButton.Location = new
System.Drawing.Point(323, 39);
            this.RecalibrateButton.Name = "RecalibrateButton";
            this.RecalibrateButton.Size = new System.Drawing.Size(134,
57);
            this.RecalibrateButton.TabIndex = 15;
            this.RecalibrateButton.Text = "Recalibrate";
            this.RecalibrateButton.UseVisualStyleBackColor = true;
            this.RecalibrateButton.Click += new
System.EventHandler(this.RecalibrateButton_Click);
            //
            // UpdateClockTimer
            //
            this.UpdateClockTimer.Interval = 250;
            //
            // ClockBox
            //
            this.ClockBox.Font = new System.Drawing.Font("Microsoft Sans
Serif", 14.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.ClockBox.Location = new System.Drawing.Point(716, 52);
            this.ClockBox.Name = "ClockBox";
            this.ClockBox.ReadOnly = true;
            this.ClockBox.Size = new System.Drawing.Size(72, 29);
            this.ClockBox.TabIndex = 16;
            this.ClockBox.Text = "30";
            this.ClockBox.TextAlign =
System.Windows.Forms.HorizontalAlignment.Center;
            //
            // RobotDemo
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.LightSlateGray;
            this.ClientSize = new System.Drawing.Size(803, 732);
            this.Controls.Add(this.ClockBox);
            this.Controls.Add(this.RecalibrateButton);
            this.Controls.Add(this.StopDemoButton);
            this.Controls.Add(this.StartDemoButton);
            this.Controls.Add(this.ConnectButton);
            this.Controls.Add(this.PrinterLabel);
            this.Controls.Add(this.DeviceIpCheckBox);
            this.Controls.Add(this.RobotLabel);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.RobotIpTextBox);
            this.Controls.Add(this.DeviceScreenBox);
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MaximumSize = new System.Drawing.Size(819, 770);
            this.MinimumSize = new System.Drawing.Size(819, 770);
            this.Name = "RobotDemo";
            this.Text = "Robot Demo";
```

```
((System.ComponentModel.ISupportInitialize)(this.DeviceScreen)).EndInit();
            this.DeviceScreenBox.ResumeLayout(false);
            this.DeviceScreenBox.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.PictureBox DeviceScreen;
        private System.Windows.Forms.TextBox RobotIpTextBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label RobotLabel;
        private System.Windows.Forms.TextBox DeviceIpCheckBox;
        private System.Windows.Forms.Label PrinterLabel;
        private System.Windows.Forms.Button ConnectButton;
        private System.Windows.Forms.Timer ScreenTimer;
        private System.Windows.Forms.Button StartDemoButton;
        private System.Windows.Forms.Button StopDemoButton;
        private System.Windows.Forms.Timer DemoTimer;
        private System.Windows.Forms.GroupBox DeviceScreenBox;
        private System.Windows.Forms.Label MouseLocation;
        private System.Windows.Forms.Button RecalibrateButton;
        private System.Windows.Forms.Timer UpdateClockTimer;
        private System.Windows.Forms.TextBox ClockBox;
    }
}
```

Appendix D: Test Harness

Harness.cs

```
using System;
using HP.DeviceAutomation.Jedi;
using RobotLibrary;

namespace RobotTestHarness
{
    class Harness
    {
        private const string pass = "TestPass”;
        static void Main(string[] args)
        {
            const string greeting =
@"|Robot Library Test Interactive Interface          |
|What would you like to do?                         |
|Select 1 - 8                                       |
|1: HPAC Secure Pull Print IRM                      |
|2: HPAC Secure Pull Print                          |
|3: Make Copy                                       |
|4: Press a button on homescreen                    |
|5: Scan to Device Memory                           |
|6: Open From Device Memory                         |
|7: Check Robot Status                              |
|8: Convert Coordinates                             |
|9: Test Coordinate finder                          |
|Everything Else: Quit                              |
-----------------------------------------------------
|Selection: ";
            if (args.Length > 1)
            {
                try
                {

                    bool quit = false;
                    while (!quit)
                    {
                        Console.Clear();

Console.WriteLine("_____________________________________________________");
                        Console.Write(greeting);
                        string selection = Console.ReadLine();
                        switch (selection)
                        {

                            case "1":
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
```

```
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        Console.WriteLine("Please enter
the auth code");
                                        string authCode =
Console.ReadLine();

myRobot.hpacPullPrintIRM(authCode);
                                    }
                                }
                                break;


                            case "2":
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        Console.WriteLine("Please enter
username.");
                                        string user = Console.ReadLine();
                                        Console.WriteLine("Please enter
password.");
                                        string password =
Console.ReadLine();
                                        myRobot.hpacPullPrint(user,
password);
                                    }
                                }
                                break;

                            case "3":
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        Console.WriteLine("Please enter
the number of copies.");
                                        string numCopy =
Console.ReadLine();
                                        myRobot.makeCopy(numCopy);
                                    }
                                }
                                break;

                            case "4":
```

```
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        Console.WriteLine("Please enter
the button that you would like to push.");
                                        string button =
Console.ReadLine();
                                        myRobot.pressHomepage(button);
                                        Console.Write("Press a key to
continue...");
                                        Console.ReadKey();
                                    }
                                }
                                break;

                            case "5":
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        Console.WriteLine("Please enter
the name for the new file");
                                        string filename =
Console.ReadLine();

myRobot.scanToDeviceMemory(filename);
                                    }
                                }
                                break;

                            case "6":
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        Console.WriteLine("Please enter
either 'all' or 'firstDoc'");
                                        string whichDoc =
Console.ReadLine();
                                        Console.WriteLine("Would you like
to delete files after? Please enter either 'yes' or 'no'");
                                        string headNod =
Console.ReadLine();
                                        if (headNod == "yes")
```

```
                                        {
myRobot.retrieveFromDeviceMemory(whichDoc, true);
                                        }
                                        else
                                        {
myRobot.retrieveFromDeviceMemory(whichDoc, false);
                                        }
                                    }
                                }
                                break;

                            case "7":
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        foreach (string status in
myRobot.getStatus())
                                        {
                                            Console.WriteLine(status);
                                        }
                                        Console.Write("Press a key to
continue...");
                                        Console.ReadKey();
                                    }
                                }
                                break;

                            case "8":
                                Console.WriteLine("Please enter your X
value");
                                string X = Console.ReadLine();
                                Console.WriteLine("Please enter your Y
value");
                                string Y = Console.ReadLine();
                                try
                                {
                                    Cartographer map = new Cartographer();
map.convertCoordinates(Double.Parse(X), Double.Parse(Y));
                                }
                                catch (ArithmeticException e)
                                {
                                    Console.WriteLine(e.Message);
                                }

                                Console.Write("Press a key to
continue...");
                                Console.ReadKey();
```

```
                                break;
                            case "9":
                                Console.WriteLine("Please enter the button
id");
                                string buttonID = Console.ReadLine();
                                using (JediWindjammerDevice device = new
JediWindjammerDevice(args[0], pass))
                                {
                                    using
(RobotLibrary.JediWindjammerRobot myRobot = new
JediWindjammerRobot(device, args[1]))
                                    {
                                        myRobot.deviceStub(buttonID);
                                    }
                                }
                                break;
                            default:
                                quit = true;
                                break;
                        }
                    }
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Console.ReadKey();
                }

            }
            else
            {
                Console.WriteLine("Please set up command line arguments in
your project properties");
                Console.Write("Press a key to continue...");
                Console.ReadKey();
            }
        }
    }
}
```