Pictor Liquid Handler Manual

Version 1.0

17/08/17 N CHAU

Table of Contents

[Introduction 3](#_Toc490734679)

[Features 3](#_Toc490734680)

[Physical Specifications 3](#_Toc490734681)

[Deck Layout 4](#_Toc490734682)

[Slide Tray 4](#_Toc490734683)

[Tip Waste 4](#_Toc490734684)

[Liquid Waste 4](#_Toc490734685)

[Reagent Reservoir 4](#_Toc490734686)

[Tip Box 4](#_Toc490734687)

[Sample Rack 4](#_Toc490734688)

[Return Rack 4](#_Toc490734689)

[Bill of Materials 5](#_Toc490734690)

[System Stack Overview 5](#_Toc490734691)

[Electrical Specifications 6](#_Toc490734692)

[Wiring Diagram 6](#_Toc490734693)

[Liquid Handler Usage Instructions 7](#_Toc490734694)

[Transferring files onto the Raspberry Pi: 8](#_Toc490734695)

[Troubleshooting 8](#_Toc490734696)

[Known Issues / Ideas 8](#_Toc490734697)

[References 9](#_Toc490734698)

[Software API 9](#_Toc490734699)

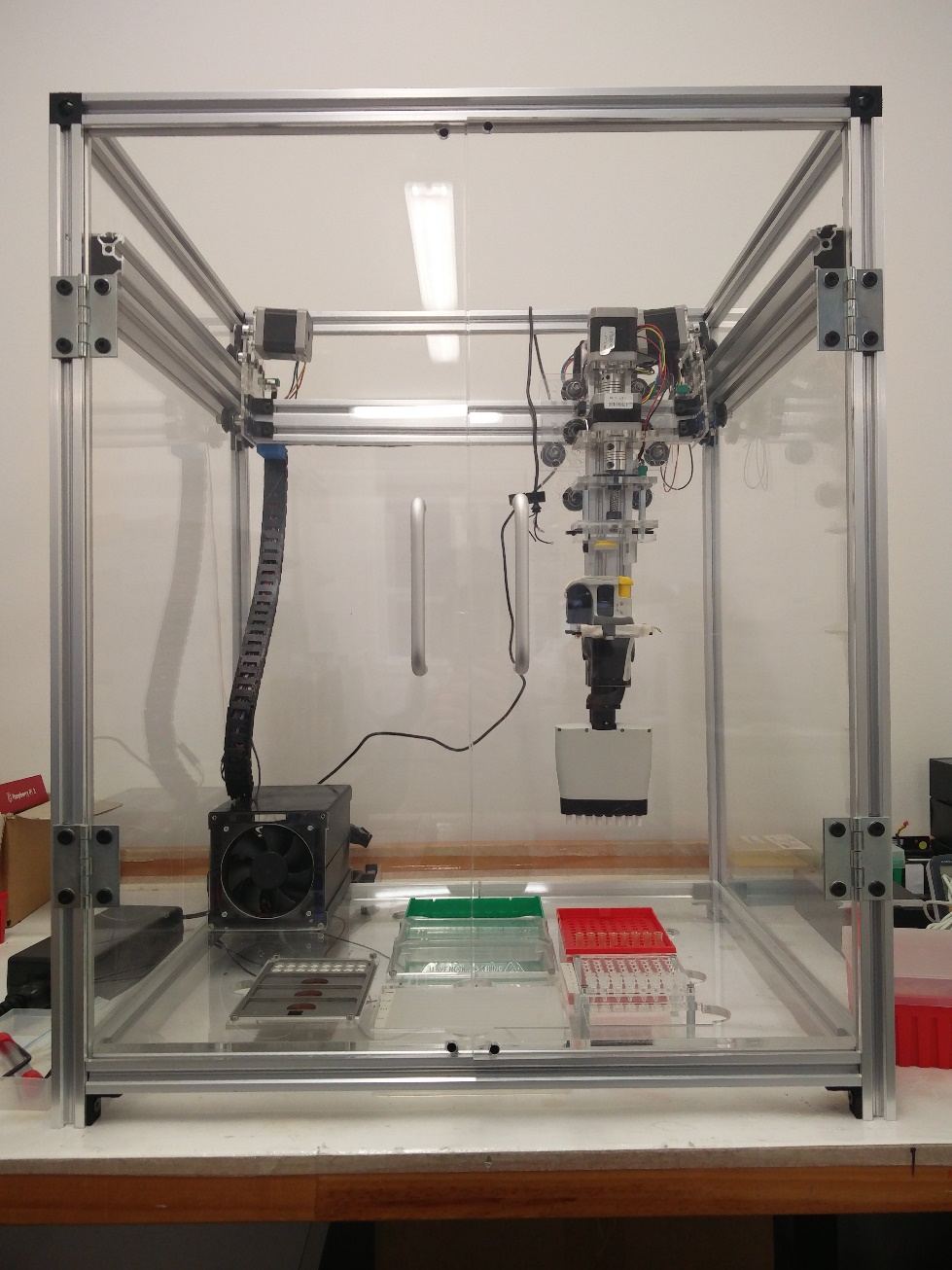
# Introduction

The Pictor Liquid Handler is an automated pipetting platform capable of processing 4 PicArray TM slides. It has a built in 37-degree heating element to incubate the samples, modular racks for sample preparation and trays to cater for standard 96 tip boxes.

# Features

* 12V 10A DC supply
* Full 4 axes control (XYZE)
* Adjustable temperature control
* Wireless communication (PC or phone controlled)
* Remote control (not yet implemented)
* Real time camera feed
* Modular racks
* Local web calibration

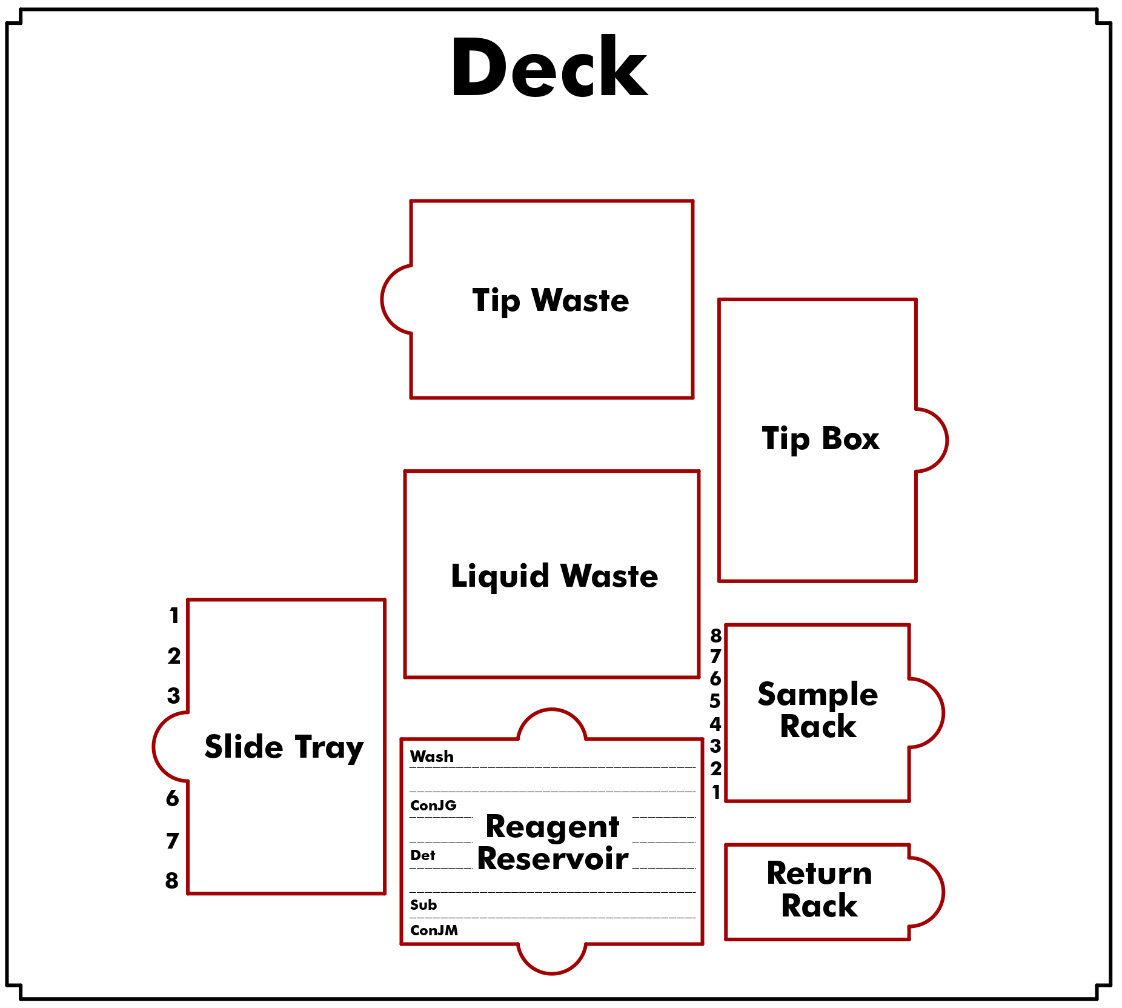
# Physical Specifications



Dimensions: 510mm x 460mm x 630mm (L/B/H)

Weight:

## Deck Layout



## Slide Tray

Aluminium tray which holds up to 4 PicArray slides which is heated up to 37-degrees during incubation

## Tip Waste

Empty 96 Tip box used to collect the tips

## Liquid Waste

50mL Biotix Disposable Reagent Reservoir used to collect all fluid waste

## Reagent Reservoir

Seahorse Reagent Reservoir used to hold reagents. Positions of each reagent is indicated in the diagram and a minimum of 3mL is required with the exception of the wash buffer which will require 5mL.

## Tip Box

Full 96 200µL Tip Box is placed into this slot

## Sample Rack

0.2mL 8-Well Strip Tubes containing the samples are placed into this detachable rack, starting from the bottom side (as indicated by the indices in the diagram)

## Return Rack

Temporary rack used to store tips which can be reused (for washing). Currently not being used.

# Bill of Materials

Refer to Bill of Materials.xlsx

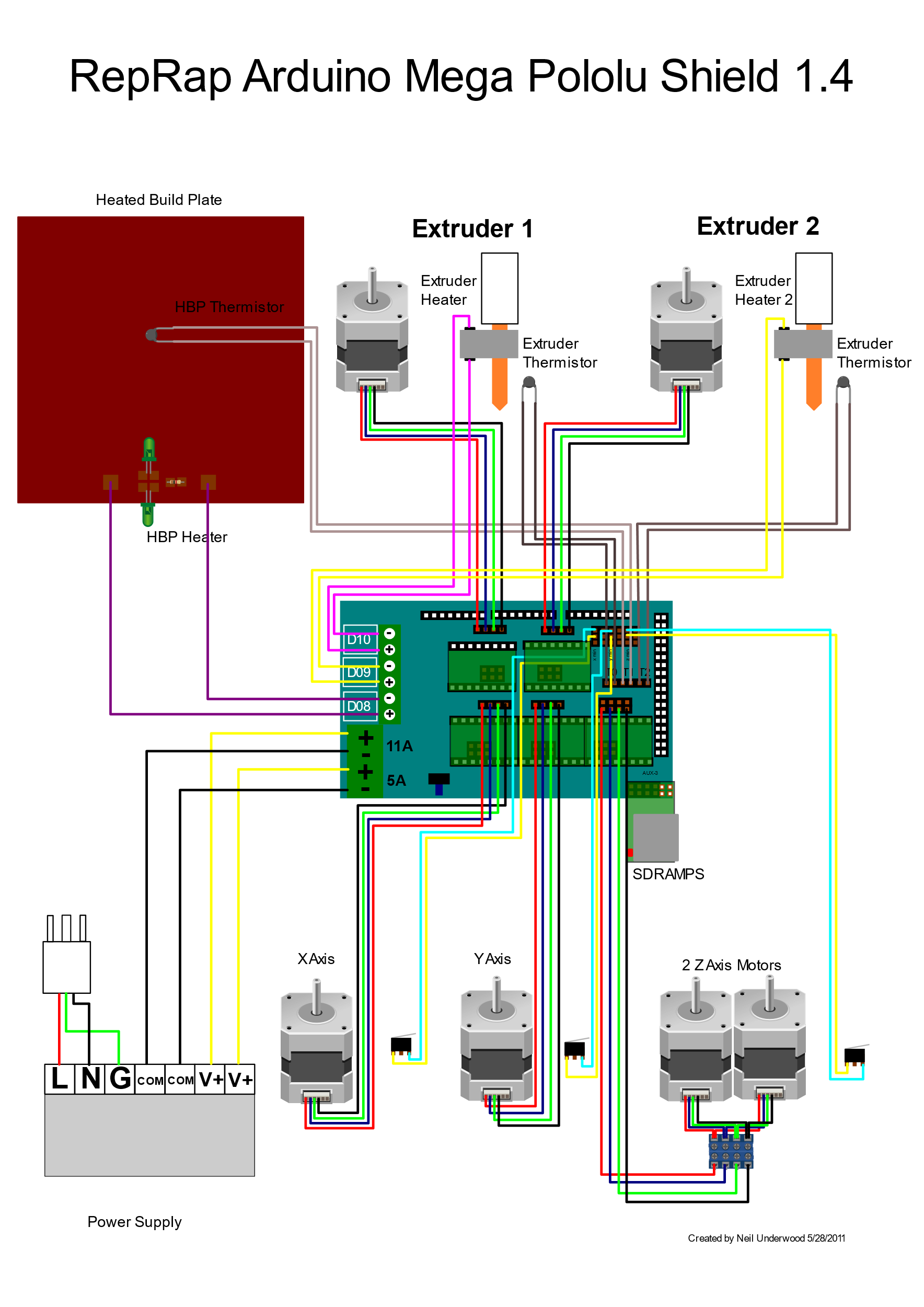
# System Stack Overview

The user sends commands through the web app built from Bootstrap 3 via HTTP GET/POST requests to the Flask server on the Raspberry Pi. The Raspberry Pi then sends G-code commands through serial between the Arduino. The Arduino then instructs the motors connected via the RAMPS 1.4 board.

A computer vision system could also be implemented in which the USB camera does live streaming of the experiment and detect if an error has occurred.

# Electrical Specifications

## Wiring Diagram



Wiring design has been adapted from the image above. The following has been modified to suit:

* 2 Y Axis motors are driven in dual mode instead of Z Axis
* Heated Build Plate is replaced with a resistive heating element
* E Axis motor wired as Extruder 1 and Extruder 2 not used
* Extruders Heaters and Thermistors removed
* D09 has been used to power a fan to cool electronics
* Serial connection established between Arduino and Raspberry Pi through USB (Arduino and Raspberry Pi not shown in this diagram)
* E Axis endstop connected into digital input pins of Raspberry Pi as a voltage divider with a 10kOhm resistor.
* 5V step-down voltage regulator used to power Raspberry Pi from 12VDC input (Arduino powered through USB from Raspberry Pi)
* USB camera connected to USB port of Raspberry Pi

# Liquid Handler Usage Instructions

Power up the machine through the switch at the back (12V 10A PSU)

Use a network scanner program (Fing) on a device on the same wireless network and search for a raspberry pi device and record its IP address. NOTE – If you cannot find the device, then connect a HDMI and keyboard directly to the raspberry pi, open a terminal and type *ifconfig*. Record the IP address located under wlan0. If there is still an error, follow the Raspberry Pi wifi setup to configure your wireless network.

Launch PUTTY (or another SSH program) and open a connection with the IP address with port 22. This should open a terminal window with the message “login as: “, the default login and password is *pi* and *raspberry* respectively.

Once logged in, navigate to the web-server directory by typing in *cd web-server*.

Start up the liquid handler web server by executing *sudo python LiquidHandlerv3.1.py* and a message should pop up saying *Running on http://0.0.0.0:80/.*

Open a web browser either on your PC or phone and enter the IP address into the URL. This should take you to the home page of the Liquid Handler.

In the root of the Raspberry Pi contains a file named serialTest.py. This script can be used to send individual G-code commands to Marlin for testing. Refer to the Marlin G-code commands list for available commands.

# Transferring files onto the Raspberry Pi:

Launch a SFTP program (FileZilla) and enter the same details as PUTTY to connect to the Raspberry Pi. This will open the file directory which will allow you to update the files to run the webserver. The webserver is hosted through Flask on Python and follows this file structure:

**web-server**

-LiquidHandler.py (Main routine script)

-coordinates.csv (File that stores the coordinates)

-camera.py (Library for camera usage)

->**templates**

-main.html (Home page)

-calibration.html

-instructions.html

->**static**

-css

-fonts

-js

-img

-favicon.ico

All files can be found in C:\Users\Pictor17\Documents\GitHub\LiquidHandler

# Troubleshooting

If for any reasons the motors are not responding, try reset the Raspberry Pi by sending a *sudo reboot* command in the PUTTY terminal.

If the motors have not homed properly, press home again, sometimes the endstops do not trigger properly. Otherwise, reset the Raspberry Pi.

Hardware configurations (motor directions/gear ratios/etc) can be altered in the Marlin Arduino sketch which will have to be reflashed onto the Arduino Mega.

## Known Issues / Ideas

* Pipette misalignment- when the pipette picks up the tip, there is a tendency for the pipette to pivot slightly which will offset the docking for the next set of tips
* Endstops mistriggering during homing
* X axis extrusion not perfectly straight due to tee nut grinding in the slot of the Y-axis extrusion
* 3D printed PLA brackets are vulnerable to snapping
* Variation in tips as well as uneven pressure of tip docking of the pipette can lead to off centred tips which may lead to missed dispenses and aspirates on the slide.
* Incubation will require a lid to prevent evaporation
* Sample dilutions have not been programmed
* Chain links need to be widened
* HDMI port needs to be more easily accessed
* Real time camera detection algorithms not yet implemented
* HTML pages require clean up and formatting
* Port forward IP address or upload server onto an internet server for remote access
* Side acrylic panels requires redesigning for easier installation
* Possible LED indicator strips for incubation/runtime/emergency stop

# References

<http://marlinfw.org/meta/gcode/>

<http://flask.pocoo.org/docs/0.12/>

<http://docs.opencv.org/3.0-beta/modules/refman.html>

<https://learn.adafruit.com/adafruits-raspberry-pi-lesson-3-network-setup/setting-up-wifi-with-occidentalis>

# Software API (Python)

**serialSend**(command)

Input: G-code command string, e.g. “G1 X100 Y100 Z100 F100”

Output: Sends command to Arduino through serial

**pickTip** (X,Y,Z,count)

Input: XYZ positions, counter for calculating rows of tips used [triggers when tip box finished at 12]

Output: Drives gantry to XYZ location and docks onto tips

**pickSample** (X,Y,count)

Input: XYZ positions, counter to determine row of sample to be picked

Output: Drives gantry to XYZ location and aspirates sample (50uL)

**pickFluid** (X,Y,E)

Input: XYZ positions, E position for volume

Output: Drives gantry to XYZ location and aspirates reagent (E volume)

**dispense** (X,Y,count,F,vol,firstFlag)

Input: XY positions, count for row position, vol for volume, Boolean firstFlag to see if it is the first dispense cycle

Output: Moves to XY position and dispenses vol

**aspirate** (X,Y,count,F,firstFlag)

Input: XY positions, count for row position, F for speed, Boolean firstFlag to see if it is the first aspirate cycle

Output: Moves to XY position and aspirates 100uL

**wash** (num,firstFlag,count)

Input: num for the iteration of wash (1,2,3), firstFlag to see if it is the first wash cycle, count for row position

Output: Macro function to pickup wash buffer, dispense, aspirate and eject

**dispose** (X,Y)

Input: XY positions

Output: Dispenses liquid into liquid waste tray

**eject** (X,Y,count)

Input: XY positions, count for row position

Output: Ejects the tips into the waste box

**homeE** ()

Input: None

Output: Homes the E axis (RAMPS cannot support 5 motors so homeE serves the homing of the pipette motor)

**stringFormat** (X = None,Y = None,Z = None, E = None,F = None)

Input: Overload XYZ positions, pipette volume E , motor speed F

Output: Constructs a G-code compatible string from the inputs

**incubation**(seconds)

Input: Time in seconds

Output: Sends G-code command to heat heating element to 40 degrees for seconds

**runProgram**()

Input: None

Output: Main routine for Liquid Handler program

**timer**(init)

Input: initial time init

Output: Elapsed time used for display and timing on GUI

**get\_incubationTime**(s)

Input: seconds s

Output: Returns the incubation time remaining for display on GUI

All the HTML request functions in LiquidHandler.py are referenced in all the HTML pages. Refer to flask and generic HTML troubleshooting guides for additional information.