Simple Cartesian Mosaic Constructor

A Manufacturing Experience

Ben Falter, Josiah Rom, Daniel Shafer, Timothy Williams

ENGR 480

Professor Stirling

Class Project Summary

13 December 2018

Introduction:

Manufacturing Class was, as it turned out, a brief exploration of many of the elements used in modern automated manufacturing. Particularly, it focused on the technologies used to control, manipulate, and interrogate the machines and processes used to create many of today's marvels. In so doing we discussed CNC control software and hardware, hardware for controlling modern machinery and inputs, and sensors to provide feedback to those systems. Therefore, it made sense that our lab time would be devoted to a project that would support all of those things; a project to create a machine that would perform an assigned task repeatedly and reliably. This year that project was to build a CNC mosaic constructor that could assemble a mosaic from tiles of four different colors, twelve tiles square. Initially our team had bullish plans for how to accomplish the sorting and feeding aspects of our assigned task, and we did a fair amount of work towards those lofty goals, but in the end we were forced to settle for just a machine that could do the end task satisfactorily. In the end our machine was able to perform somewhat reliably, though it would have been much better if we could have implemented our sorting and feeding design so as to allow it to be more autonomous than it ended up being.

Instructions:

Loading the machine

We did not get to build the gravity feed hopper and sorting mechanism that would have fed the tiles in the correct orientation to the base. As such, tiles have to be fed manually into the track (Figure 4) with the glazed side up. Once there are a few tiles on the track, then make sure that they have gone all the way down and are fully against the end of the track near the sensors.

Starting the machine

For starting the machine all related files must be loaded onto the control laptop, 24 V power and 5 V power must be connected, and the bed must be clear. The starting procedure is:

1. Verify E Stop is disengaged.
2. Load G-code file into Linux CNC.
3. Turn off electric E Stop (red x).
4. Activate power in Linux CNC (red power button).
5. Home x-axis, Home y-axis
6. Trigger program start in Linux CNC

Clearing jams

Depending on the cause of the jam, the procedure for clearing the jam is:

A. Picker looping through pick up procedure (without a tile)
    1. Verify that the feed ramp is not over filled by referencing fill lines
    2. Attempt to loosen tile in position
    3. If unable to free activate E stop and clear run (physical and Electronic)
    4. Refer to vacuum failure section below
B. Picker looping through pick up procedure (with a tile)
    1. Verify that the feed ramp is not over filled by referencing fill lines
    2. Remove tile from suction and allow to pick up a new tile

3. If unable to free activate E stop and clear run (physical and Electronic)
4. Refer to sensor failure section below

C. No tile feeding into pick up location
1. Check tile dimensions, inconsistent tiles can jam in feeder (basic presort required)
2. Clear run and reset system

Operation

Linux CNC

The system for controlling the mosaic tile building robot is Linux CNC. It reads G-Code and interprets it into motion for cartesian motion. Our system is driven by stepper motors rotating lead screws for determining linear motion. For this project the G-Code is created using a text file for the colors and MATLAB looping through the same commands and moving position. The relevant MATLAB code can be seen below.

Feeding System

For feeding the tiles a ramp was designed and it uses gravity to force the tiles into the pick-up section. If the system is over filled the backpressure fights the vacuum pick up and causes failure in operation. A fill line is clearly marked on the ramp for reference. Also, if the tile que falls below the empty line the tiles will not have enough force to be feed into the pick-up location. Again, an empty line is clearly marked on the ramp. Future work talks about our plan for keeping the que between these two marks.

Vacuum Pick-up

The system for picking up the tiles uses compressed air forced though a venturi vacuum system that has a soft rubber end for creating a seal with the top of the smooth tiles. Pistons are used to lower and raise the tiles grabbed by the vacuum.

Cartesian Motion

Once the tiles are lifted out of the pick-up location they need to be place on the bed. For this stepper motor driven slides are moved from Linux CNC. The generated G-Code gives directions for where the stopping points for the placement of each tile should be, and the pick-up location is dependent on the required color of tile.

Setting Up the Bed

For setting up the bed for placing the tiles a cloth mesh needs to be placed and oriented based on where the tiles are going to be placed. For reference on the placement of the mesh look for the bed marks that show the outside corners of a finished mosaic. About a 1/8 extra is needed past each side to allow for holding down the mesh.

Removing the Final Mosaic

Once a cycle has completed the final mosaic needs to be removed. For this place the catching tray up against the bed and detach the mesh. Slide the tile covered mesh slowly onto the tray and carry to next stage of the operation.

Maintenance

Sensor Failure:

Symptoms – tiles are not noticed to have been picked up and will just keep mashing tiles up and down.

Fix – move a tile to the very end of the track. Move the sensor to a point where it does not recognize the tile. Then adjust sensitivity to find the tile, and still trigger at half a tile length.

Slider Failure:

Symptoms – either the bed does not move upon completion of a row, or the vacuum does not move to pick and place tiles.

Fix – Take the slider apart by undoing the screws near the motor. Then check that the connection between the motor and the lead screw is still solid. If the connection is solid, then you may need to just replace the slider assuming cords have been checked to still be connected correctly. To remount, use t-nuts to bolt into a horizontal position on the two supports at either end. Then make sure the height is such that at either end of the pneumatic stroke can reach the tiles, and clear of any obstacles between the bed and the track. Reattach the homing and driving wires to their respective cables.

Track Failure:

Symptoms – the track is broken or bent into an unwanted shape

Fix – unscrew the portion that is faulty, reprint a new track part, sand down any rough edges and screw into place.

Vacuum Failure:

Symptoms – the tile is not ever getting picked up.

Fix – check the tube providing air to the vacuum, if there is air flow present, then replace the venturi vacuum pump. If there is no flow, check to see if there are leaks in the cable anywhere, or possibly if the cable is cut or broken anywhere. If broken, then replace the tube.

Pneumatic Failure:

Symptoms – the vacuum is not getting moved up and down at the pick and place locations

Fix – check the air tube to see is air is flowing, and that there are not breaks in the hose. If there are, replace the hose. If not, then the pneumatic may need to be replaced. This can be accomplished by undoing the air tubes, unbolting the vacuum and slider, and then switching it to a new pneumatic.

Suggestions for future improvements

In the course of this project we were initially rather bullish with regards to what we thought we could accomplish. Indeed, the minimum requirements for the project left a lot of room for

improvement, but we were nevertheless unsuccessful in doing anything beyond the basics. However, this was not entirely our fault, for as it happened we needed to use a large number of printed parts, and as it turned out the printer was having issues correctly printing our parts. Several of the parts we tried to print ended up being warped beyond reason and had to be discarded, and at other times the prints would skew while printing, producing a mess of almost unrecognizable material. On a few occasions (during the Thanksgiving break) we were able to print our parts with no one else around, and we got very good results, by and large. One print warped somewhat, but it was removed before being allowed to fully cool on the print bed. The parts that were undisturbed and left to cool fully on the bed turned out well. Therefore it is possible that many of the warpage incidents were due to people in the lab disturbing the prints before they were fully cooled. That said, our future improvements list contains both the items we were trying to implement as well as some that were based on the results we observed in our "completed" project.

One of the most significant improvements that could be made to our machine would be a tile feeder. Hand feeding our machine was tedious, and if our machine had been much faster we would not have been able to keep up with it. Also, hand feeding defeats much of the purpose of automating the production process – to remove as many greedy employees from the process as possible. Increasing the rate of work completion is not as valuable as doing so while removing the manpower requirement. Indeed, an optimal manufacturing system would have no employees at all, especially in today's world with the greed of unskilled laborers wanting more pay than their work is often worth. To that end, we had originally meant for our system to be able to operate autonomously enough that it could be filled with un-oriented tiles via a hopper and then fed via an orienting and feeding system. As envisioned, the feeding system was to consist of four hoppers full of tiles, a rotary removal device, and a sorting system. The rotary device was supposed to be mounted with a fine tolerance fit into the side of the hopper so as to acquired tiles from below and prevent jams. Then the tiles were to be sent into a single sorting device that would meter, orient, and finally sort the tiles by color. The feeding system from the hoppers was meant to only operate when needed, and individually by required color instead of all at once. Once through the orienting system, the tiles were to be sent to a one way in, four way out sorting gate consisting of a single entry point, translatable and skewable guide walls, a small linear motor, and four exit points. From here the tiles would then go directly to the queue.

Integral to our feeding system was the orienting system, which was meant to determine if tiles were appropriately oriented for placement. The tiles, being generally bi-stable with only two normally stable orientations while undergoing acceleration while subject to multiple forces, need to be oriented so that their glazed side faces upward for placement. Therefore, determining whether a tile is glazed side up or glazed side down is the only real problem, and we noticed that the glazed side of the tiles slid much more easily on most surfaces as compared to the unglazed side. To exploit this we designed a system that would allow the tiles to build up kinetic energy and then be subjected to a test. The test was to be a flat stretch of the tile's path over which a tile with the glazed side down would slide completely to the end with velocity to spare, but which would fully retard the velocity of a tile with the glazed side up before reaching the end. The correctly oriented tiles, with their glazed side up, would then be pushed off onto a side ramp leading to the sorting system. Tiles with their glazed side down would enter a flipping track before going to the sorting system. This system, along with the rest of the feeding system would have been able to address many of the issues we encountered with regards to picking up tiles from the queue, and it would have allowed our system to be able to operate more autonomously.

However, this system was not implemented and only marginally tested, so it is likely that many modifications would have had to have been made to it as well.

In the course of testing our resulting system we found an issue with sensing that would have been best addressed via a modification to how we sensed whether a tile was picked up or not. In the system we created we had four optical proximity sensors mounted in the ends of each tile queue path, with the intent being that we would be able to detect if a tile had been picked up or not by whether we detected the brief period during which the picked up tile was missing and the one behind it had not yet slid forward to take its place. However, we encountered problems with false positives when a tile was picked up partially but became stuck above the sight of the sensor. To address this issue we decided that it would be most effective to simply remove all the proximity sensors an instead use a single light of sight sensor mounted off-angle to the y-axis (the axis that the pickup vacuum moved on) that would detect whether the section cup has a tile suspended below it or not. This sensor would be mounted to detect tiles when the pickup system was retracted, and enough off-angle to not be in the way of carriage during translation while at the same time not so much that would not be able to detect tiles in all of the four possible positions.

Other improvements could be made, but feeding, orientation, and better sensing are the most important ones at this point. Perhaps flipping the desired orientation would make much of task easier due to the lower friction involved, but doing so would probably also introduce new problems. Thus it is the fate of all manufacturing systems to need fine tuning and skilled engineers to maintain and improve them. We are ever working toward making the unskilled jobless and allowing only a few skilled individuals to take their place.

Linux CNC

Since this project requires a series of the same commands over and over for each location in the 12 x 12 grid MATLAB code was written that could loop through and write each with just the original commands. The MATLAB code can read the color and position from a text file, store position variables, and print the G-code to a text file. This text file is then transferred to the control laptop and Linux CNC reads it and follows the program.

Appendix 1

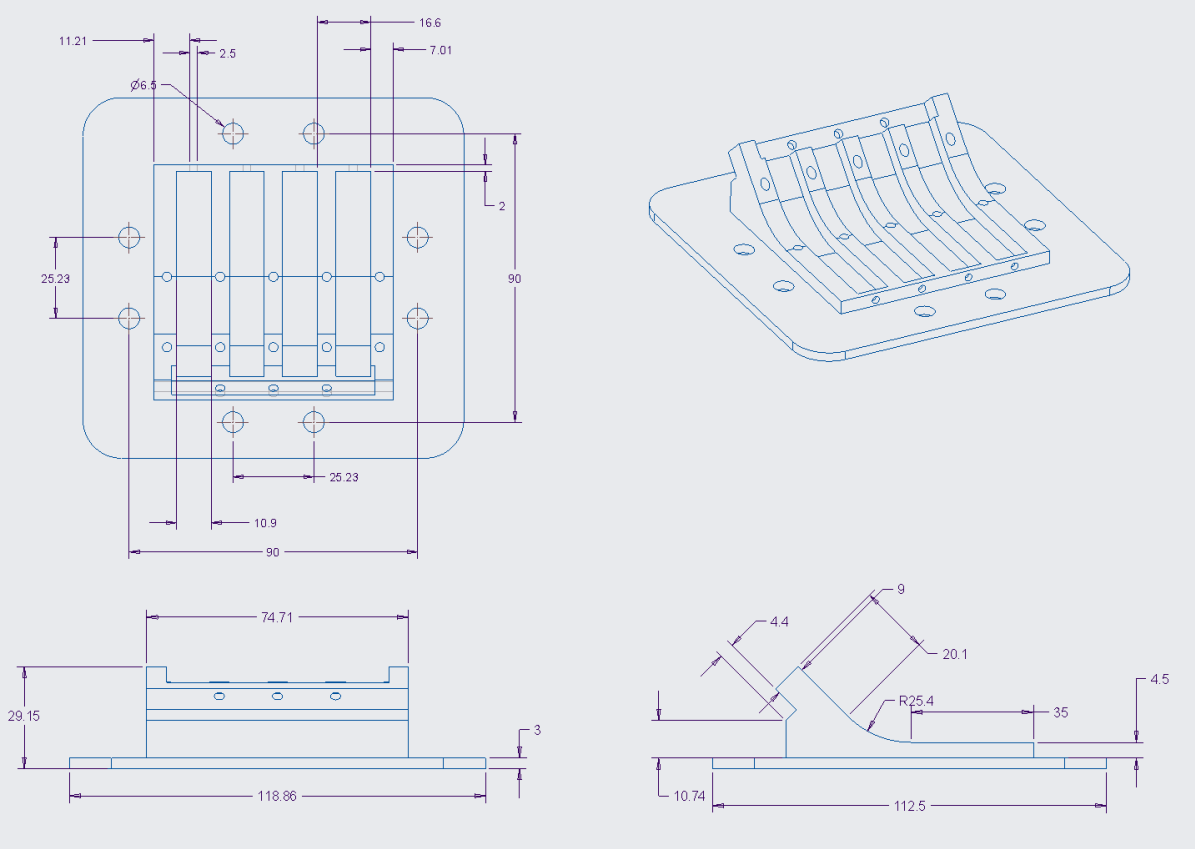


Figure 1. Queue pickup point

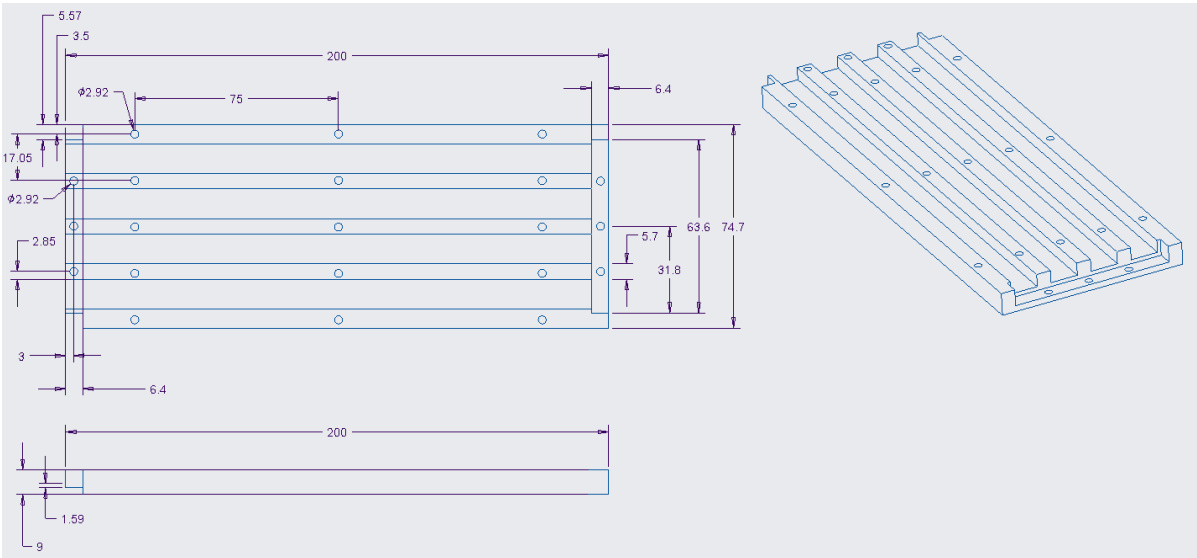


Figure 2. Entry track

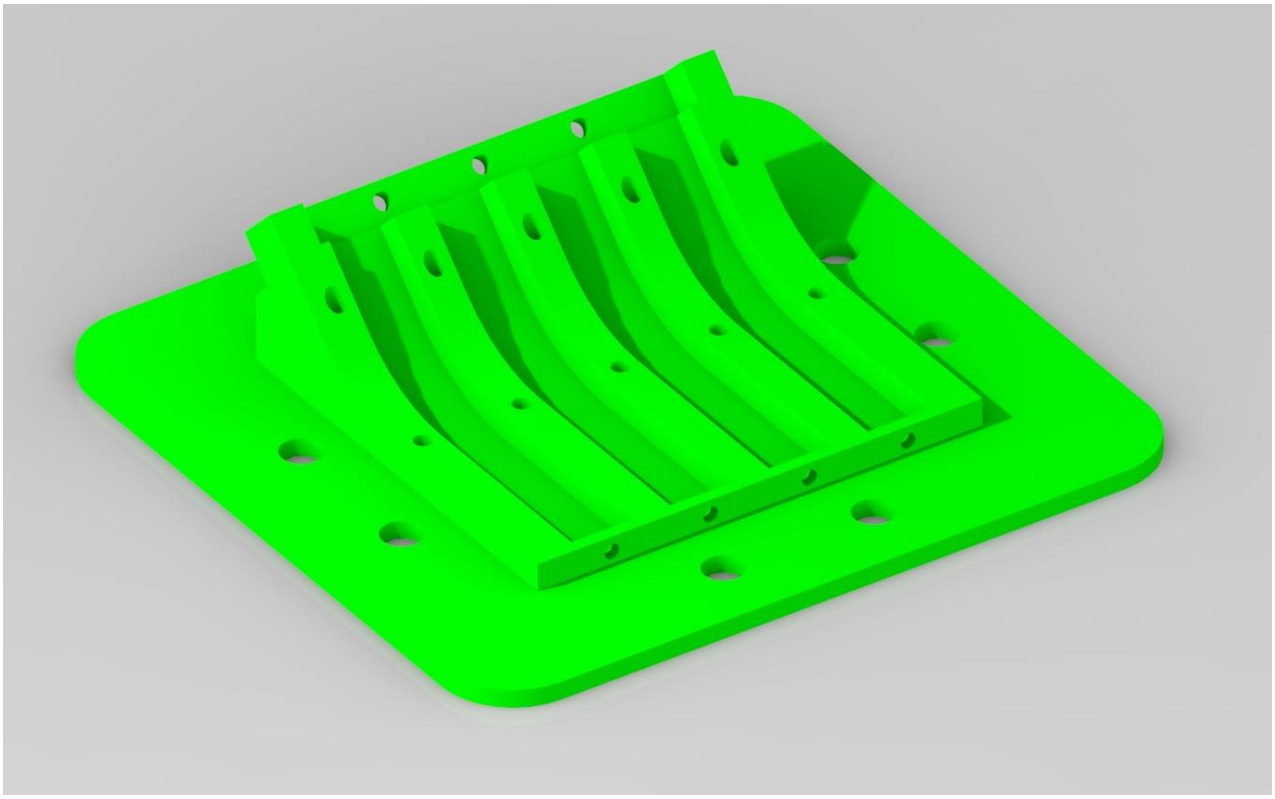


Figure 3. Rendered queue pickup point

Figure 4. Rendered entry track

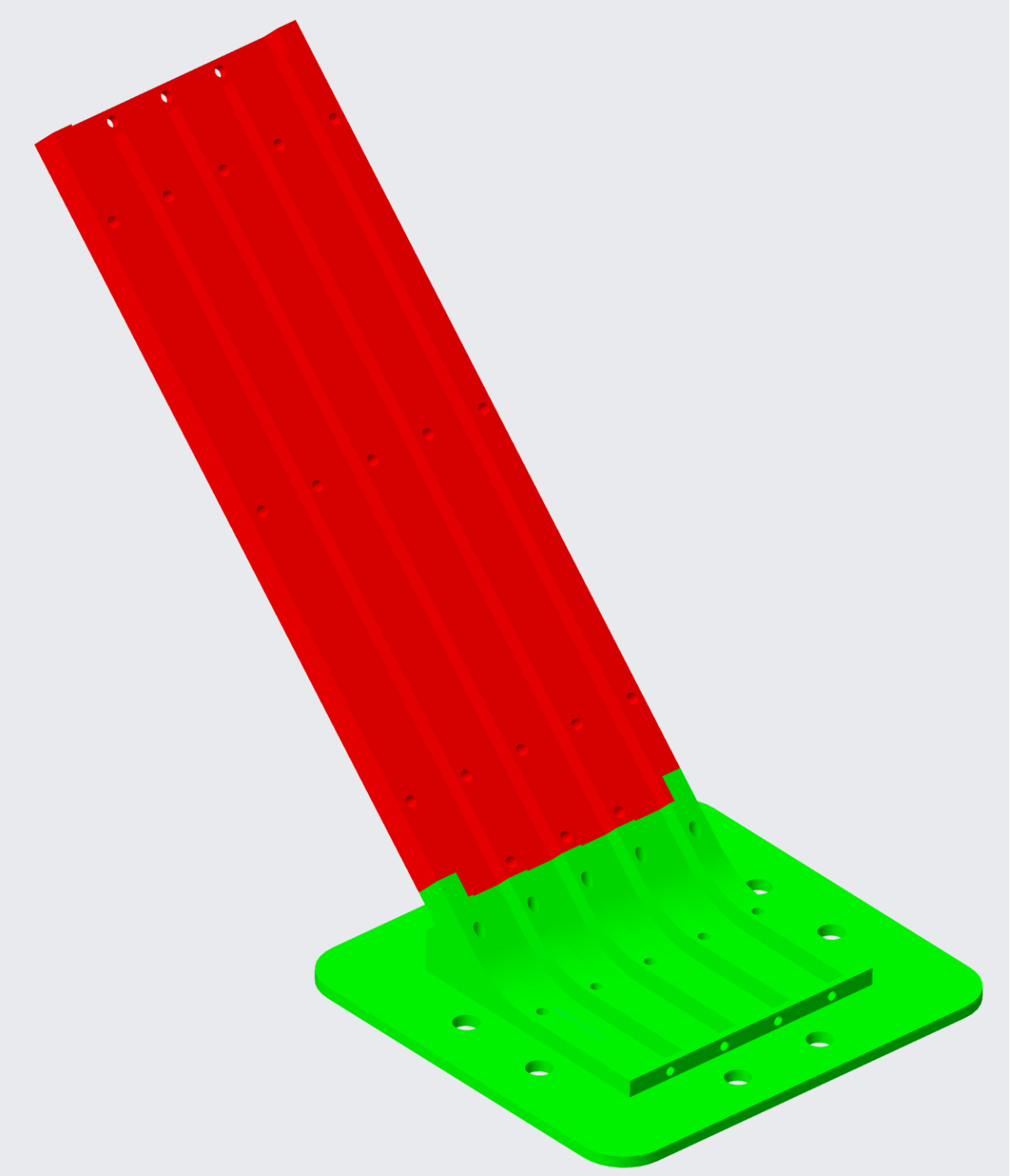


Figure 4. Overall queueing assembly at completion of project requirements

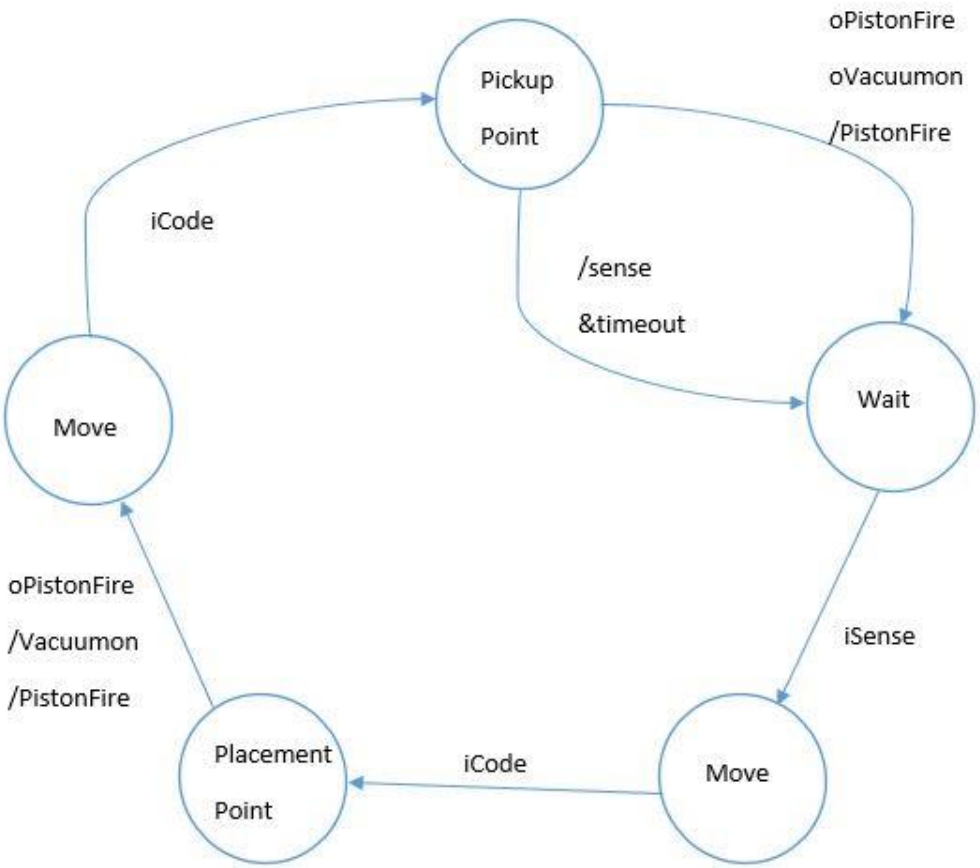
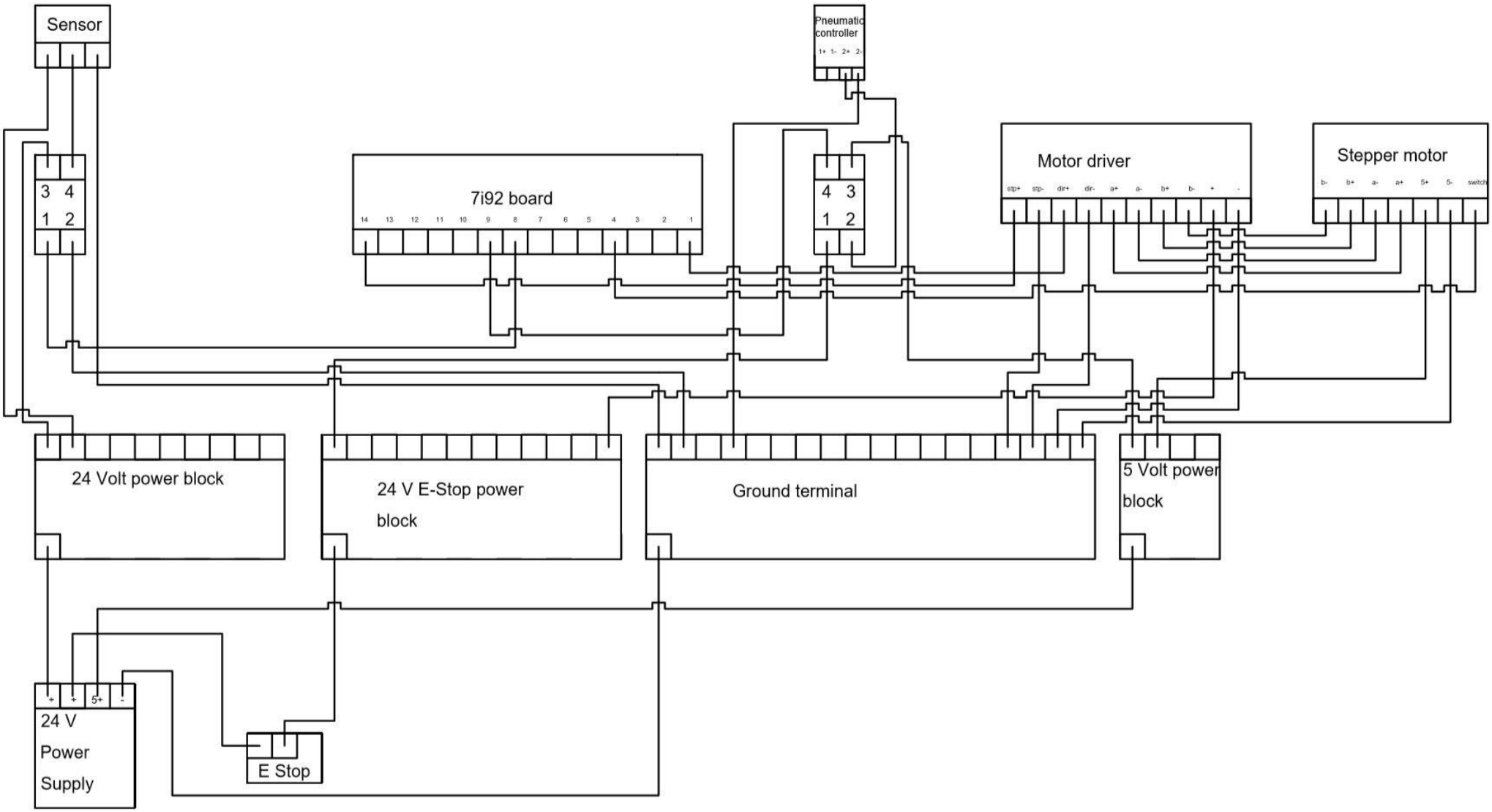


Figure 5. State Diagram for operation of our machine


Sensor
3 4
1 2
7i92 board
14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pneumatic controller
1+ 1- 2+ 2-
4 3
1 2
Motor driver
stp+ stp- dir+ dir- a+ a- b+ b- + -
Stepper motor
b- b+ a- a+ 5+ 5- switch
24 Volt power block
24 V E-Stop power block
Ground terminal
5 Volt power block
+ + 5+ -
24 V Power Supply
E Stop

Wire	From	To
24 V Power cable 1	24 V Power Supply positive	E Stop
24 V Power cable 2	24 V Power Supply positive	24 V Power Block
24 V Ground	24 V Power Supply Ground	Ground Block
Relay 6 Power Supply	24 V Power Block	Relay 6 Port 1
Relay 5 Power Supply	25 V Power Block	Relay 5 Port 1
Relay 6 Pneumatic Control	Relay 6 Port 2	Pneumatic Control Port 4
Pneumatic Control 4 Ground	Pneumatic Control Port 4	Ground Block
Relay 5 Pneumatic Control	Relay 5 Port 2	Pneumatic Control Port 3
Pneumatic Control 3 Ground	Pneumatic Control Port 3	Ground Block
Sensor Power Supply	24 V Power Block	Sensor Power Block
Sensor 4 Power	Sensor Power Block	Sensor 4
Sensor 3 Power	Sensor Power Block	Sensor 3
Sensor 2 Power	Sensor Power Block	Sensor 2
Sensor 1 Power	Sensor Power Block	Sensor 1
Sensor 4 Data	Sensor 4	Relay 1 Port 4
Sensor 3 Data	Sensor 3	Relay 2 Port 4
Sensor 2 Data	Sensor 2	Relay 3 Port 4
Sensor 1 Data	Sensor 1	Relay 4 Port 4
Relay 1 Positive Power	Sensor Power Block	Relay 1 Port 3
Relay 2 Positive Power	Sensor Power Block	Relay 2 Port 3
Relay 3 Positive Power	Sensor Power Block	Relay 3 Port 3
Relay 4 Positive Power	Sensor Power Block	Relay 4 Port 3
Relay 1 Output	Relay 1 Port 1	7i92 Port 11
Relay 2 Output	Relay 2 Port 1	7i92 Port 6
Relay 3 Output	Relay 3 Port 1	7i92 Port 7
Relay 4 Output	Relay 4 Port 1	7i92 Port 8
Relay 1 Ground	Relay 1 Port 2	Ground Block
Relay 2 Ground	Relay 2 Port 2	Ground Block
Relay 3 Ground	Relay 3 Port 2	Ground Block
Relay 4 Ground	Relay 4 Port 2	Ground Block
Y step Power	5 V Power Block	Y step motor Power
X step Power	5 V Power Block	X step motor Power
Y step Ground	Ground Block	Y step motor ground
X step Ground	Ground Block	X step motor ground
X step Control	X Step Control	7i92 Port 4
Y Step Control	Y step Control	7i92 Port 17
Relay 5 Power	5 V Power Block	Relay 5 Port 3
Relay 6 Power	5 V Power Block	Relay 6 Port 3
Relay 5 Input	Relay 5 Port 4	7i92 Port 10
Relay 6 Input	Relay 6 Port 4	7i92 Port 9
7i92 Port 18 Ground	7i92 Port 18	Ground Block
7i92 Port 19 Ground	7i92 Port 19	Ground Block
7i92 Port 20 Ground	7i92 Port 20	Ground Block

7i92 Port 21 Ground	7i92 Port 21	Ground Block
X Dir +	7i92 Port 1	X Step Driver Dir +
Y Dir +	7i92 Port 2	Y Step Driver Dir +
Y Step +	7i92 Port 15	Y Drive Step +
X Step +	7i92 Port 14	X Drive Step +
X Dir -	X Dir -	Ground Block
Y Dir -	Y Dir -	Ground Block
X Step -	X Step -	Ground Block
Y Step -	Y Step -	Ground Block
X Driver Power	X Step Drive Power	24 V E Stop
Y Driver Power	Y Step Drive Power	24 V E Stop
X Driver Ground	X Step Drive Ground	Ground Block
Y Driver Ground	Y Step Drive Ground	Ground Block
Sensor 1 Ground	Sensor 1 Ground	Ground Block
Sensor 2 Ground	Sensor 2 Ground	Ground Block
Sensor 3 Ground	Sensor 3 Ground	Ground Block
Sensor 4 Ground	Sensor 4 Ground	Ground Block

Figure 6. Wiring diagram and legend

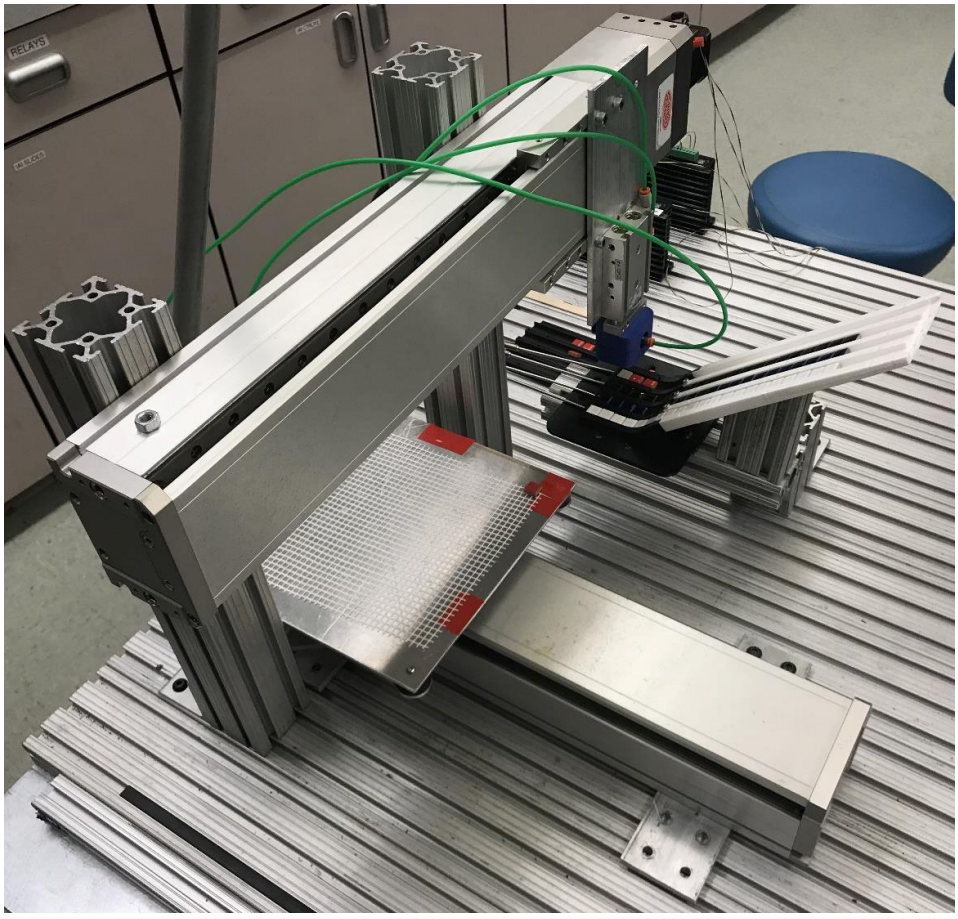


Figure 7. Overall setup

Total Errors	Total Tiles Placed
42	432
Error Percentage	9.72%
Issue	# of Errors
Did not pick up	15
Did not pick up but continued	4
Feed Jam	13
Knocked next tile loose	1
Picked up and dropped	5
Picked up and kept trying	4

Table 1. Performance data summary

Appendix 2

MATLAB Code:

```
%ENGR 480 Project
%Josiah Rom, Timothy Williams, Ben Falter, Daniel Shafer
%12/12/2018
%All units mm

close all
clc

%Reads the text file with the directions for tiles.
%Places them in a 12x12 matrix
C = dlmread('ColorFile.txt');
TestData = fopen('G-Code.txt','w');

%Initial Positions to be changed based on final design.
X0 = 150.0; %X coordinates for P (1,1)
Y0 = 20.0; %Y coordinates for P (1,1)

step = 12.7; %How Far the CNC needs to move for each new tile

X1 = 2.128; %X coordinates for color 1 (red)
X2 = 18.828; %X coordinates for color 2 (black)
X3 = 35.028; %X coordinates for color 3 (blue)
X4 = 52.128; %X coordinates for color 4 (white)

%CNC program specifications
Motion = 3500; %Feed Rate for Motion of Motors
Dwell = 0.1; %Dwell for M64
SDwell = 0.05; %Dwell for M65
SenseCheck = 0.4; %Time M66 Looks for Input

%Pin Outs for Air Control
VacuumOn = 0; VacuumOff = 1; PistonFire = 2; PistonRetract = 3;

%Sensor Pins
S1 = 3; S2 = 2; S3 = 1; S4 = 0;

%Labels New Section Titles
Olabel1 = 101; Olabel2 = 100;

%Building the Matrix
Data = zeros(3,144);
count = 1;

%Counter for Data Loop
%Fills the Data Matrix
for col = 1:12
```

```
        for row = 1:12
                Data (1, count) = X0+(row-1) *step;   %1 row - x coordinates
                Data (2, count) = Y0+(col-1) *step;   %2 col - y coordinates
                Data (3, count) = C (col, row);       %3 color for that position
                count = count + 1;
        end
end

%Code to Generate the G-code Segments for the Given Design
%Pre Written G-Code for Housekeeping
fprintf(TestData,'(AXIS,stop)\r\n'); %Tells Linux CNC Not to Draw
fprintf(TestData,'#5399 = 1\r\n');
for r = 1:144 %Each case looks at the color and determines where to pick it up from.
        if Data(3,r) == 1
                Output = X1; Sensor = S1; %Sensor 1
        end
        if Data(3,r) == 2
                Output = X2; Sensor = S2; %Sensor 2
        end
        if Data(3,r) == 3
                Output = X3; Sensor = S3; %Sensor 3
        end
        if Data(3,r) == 4
                Output = X4; Sensor = S4; %Sensor 4
        end

        fprintf(TestData,'\r\n');
        fprintf(TestData,'\r\n');
        fprintf(TestData,'G1 X%f Y%f F%f \r\n',Output,Data(2,r),Motion); %Go to Pick up
        fprintf(TestData,'M64 P%d \r\n',VacuumOn); %Turn on Vacuum
        fprintf(TestData,'G4 P%f \r\n',Dwell); %Wait
        fprintf(TestData,'M65 P%d \r\n',VacuumOn); %Deactivate
        fprintf(TestData,'G4 P%f \r\n',SDwell); %Wait
        fprintf(TestData,'\r\n');
        fprintf(TestData,'#1100 = 1\r\n'); %Set Variable for If Statement
        fprintf(TestData,'\r\n');
        fprintf(TestData,'o%d while [#1100 EQ 1]\r\n',Olabel1); %Begin While Loop
        fprintf(TestData,'M64 P%d \r\n',PistonFire); %Fire Piston
        fprintf(TestData,'G4 P%f \r\n',Dwell); %Wait
        fprintf(TestData,'M65 P%d \r\n',PistonFire); %Deactivate
        fprintf(TestData,'G4 P%f \r\n',SDwell); %Wait
        fprintf(TestData,'M64 P%d \r\n',PistonRetract); %Retract Piston
        fprintf(TestData,'M66 P%d L1 Q%f\r\n',Sensor,SenseCheck); %Look for Sensor Input
        fprintf(TestData,'G4 P%f \r\n',Dwell); %Wait
        fprintf(TestData,'M65 P%d \r\n',PistonRetract); %Deactivate
        fprintf(TestData,'\r\n'); fprintf(TestData,'o%d if [#5399 NE -1]\r\n',Olabel2); %Begin If
        Check
        fprintf(TestData,'#1100 = 2\r\n'); %Break from While
        fprintf(TestData,'o%d end if\r\n',Olabel2); %End If Check
        fprintf(TestData,'\r\n');
        fprintf(TestData,'o%d end while\r\n',Olabel1); %End While Loop
```

```
        fprintf(TestData,'\r\n');
        fprintf(TestData,'G1 X%f Y%f F%f \r\n',Data(1,r),Data(2,r),Motion); %Go to P(x,y)
        fprintf(TestData,'M64 P%d \r\n',PistonFire); %Fire Piston
        fprintf(TestData,'G4 P%f \r\n',Dwell); %Wait
        fprintf(TestData,'M65 P%d \r\n',PistonFire); %Deactivate
        fprintf(TestData,'G4 P%f \r\n',SDwell); %Wait
        fprintf(TestData,'M64 P%d \r\n',VacuumOff); %Turn off Vacuum
        fprintf(TestData,'G4 P%f \r\n',Dwell); %Wait
        fprintf(TestData,'M65 P%d \r\n',VacuumOff); %Deactivate
        fprintf(TestData,'G4 P%f \r\n',SDwell); %Wait
        fprintf(TestData,'M64 P%d \r\n',PistonRetract); %Retract Piston
        fprintf(TestData,'G4 P%f \r\n',Dwell); %Wait
        fprintf(TestData,'M65 P%d \r\n',PistonRetract); %Deactivate
        fprintf(TestData,'G4 P%f \r\n',SDwell); %Wait
        Olabel1 = Olabel1 +2; %Increase section labels
        Olabel2 = Olabel1-1;
end

fprintf(TestData,'G1 X10.0 Y280.0 F1000\r\n');
fprintf(TestData,'M30\r\n'); %End Program

%Close G-Code File
fclose(TestData);
```

G-Code Example:

```
(AXIS, stop)
#5399 = 1

G1 X2.128 Y20.0F3500.0
M64 P0
G4 P0.10
M65 P0
G4 P0.050

#1100 = 1

o101 while [#1100 EQ 1]
        M64 P2
        G4 P0.10
        M65 P2
        G4 P0.050
        M64 P3
        M66 P3 L1 Q0.40
        G4 P0.10
        M65 P3

        o100 if [#5399 NE -1]
                #1100 = 2
```

o100 end if

o101 end while

```
G1 X150.0 Y20.0 F3500.0
M64 P2
G4 P0.10
M65 P2
G4 P0.050
M64 P1
G4 P0.10
M65 P1
G4 P0.050
M64 P3
G4 P0.10
M65 P3
G4 P0.050
```

Color Text File:

1	3	3	3	4	1	1	4	3	3	3	1
4	1	3	3	4	1	1	4	3	3	1	4
3	4	1	3	4	1	1	4	3	1	4	3
3	3	4	1	4	1	1	4	1	4	3	3
4	4	4	4	4	1	1	4	4	4	4	4
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	1	1	4	4	4	4	4
3	3	4	1	4	1	1	4	1	4	3	3
3	4	1	3	4	1	1	4	3	1	4	3
4	1	3	3	4	1	1	4	3	3	1	4
1	3	3	3	4	1	1	4	3	3	3	1

Excel Plot Example:

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	3	3	3	4	1	1	4	3	3	3	1
2	4	1	3	3	4	1	1	4	3	3	1	4
3	3	4	1	3	4	1	1	4	3	1	4	3
4	3	3	4	1	4	1	1	4	1	4	3	3
5	4	4	4	4	4	1	1	4	4	4	4	4
6	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1
8	4	4	4	4	4	1	1	4	4	4	4	4
9	3	3	4	1	4	1	1	4	1	4	3	3
10	3	4	1	3	4	1	1	4	3	1	4	3
11	4	1	3	3	4	1	1	4	3	3	1	4
12	1	3	3	3	4	1	1	4	3	3	3	1

Colors			
4	3	2	1
48	36	0	60

Program ini File:

```
# Generated by PNCconf at Mon Nov 19 09:02:55 2018
# If you make changes to this file, they will be
# overwritten when you run PNCconf again

[EMC]
MACHINE = DEWEY2
DEBUG = 0

[DISPLAY]
DISPLAY = axis
POSITION_OFFSET = RELATIVE
POSITION_FEEDBACK = ACTUAL
MAX_FEED_OVERRIDE = 2.000000
MAX_SPINDLE_OVERRIDE = 1.000000
MIN_SPINDLE_OVERRIDE = 0.500000
INTRO_GRAPHIC = linuxcnc.gif
INTRO_TIME = 5
PROGRAM_PREFIX = /home/mfglab/linuxcnc/nc_files
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm .005mm
POSITION_FEEDBACK = ACTUAL
DEFAULT_LINEAR_VELOCITY = 6.000000
MAX_LINEAR_VELOCITY = 75.000000
MIN_LINEAR_VELOCITY = 0.500000
DEFAULT_ANGULAR_VELOCITY = 12.000000
MAX_ANGULAR_VELOCITY = 180.000000
MIN_ANGULAR_VELOCITY = 1.666667
```

```
EDITOR = gedit
GEOMETRY = xyz

[FILTER]
PROGRAM_EXTENSION = .png,.gif,.jpg Greyscale Depth Image
PROGRAM_EXTENSION = .py Python Script
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
py = python

[TASK]
TASK = milltask
CYCLE_TIME = 0.010

[RS274NGC]
PARAMETER_FILE = linuxcnc.var

[EMCMOT]
EMCMOT = motmod
COMM_TIMEOUT = 1.0
COMM_WAIT = 0.010
#SERVO_PERIOD = 1000000
SERVO_PERIOD = 500000

[HOSTMOT2]
# **** This is for info only ****
# DRIVER0=hm2_eth
# BOARD0=7i92

[HAL]
HALUI = halui
HALFILE = DEWEY2.hal
HALFILE = custom.hal
POSTGUI_HALFILE = postgui_call_list.hal
SHUTDOWN = shutdown.hal

[HALUI]

[TRAJ]
AXES = 3
COORDINATES = X Y Z
LINEAR_UNITS = mm
ANGULAR_UNITS = degree
CYCLE_TIME = 0.010
DEFAULT_VELOCITY = 2.50
MAX_LINEAR_VELOCITY = 25.00
NO_FORCE_HOMING = 1
```

```
[EMCIO]
EMCIO = io
CYCLE_TIME = 0.100
TOOL_TABLE = tool.tbl

#********************
# Axis X
#********************
[AXIS_0]
TYPE = LINEAR
HOME = 0.0
FERROR = 10.0
MIN_FERROR = 1.0
MAX_VELOCITY = 75.0
MAX_ACCELERATION = 750.0
# The values below should be 25% larger than MAX_VELOCITY and MAX_ACCELERATION
# If using BACKLASH compensation STEPGEN_MAXACCEL should be 100% larger.
#STEPGEN_MAXVEL = 31.25
STEPGEN_MAXVEL = 60
STEPGEN_MAXACCEL = 937.50
P = 1000.0
I = 0.0
D = 0.0
FF0 = 0.0
FF1 = 1.0
FF2 = 0.0
BIAS = 0.0
DEADBAND = 0.0
MAX_OUTPUT = 0.0
# these are in nanoseconds
DIRSETUP   = 14000
DIRHOLD    = 14000
STEPLEN    = 14000
STEPSPACE  = 14000
STEP_SCALE = 200.0
MIN_LIMIT = -0.01
MAX_LIMIT = 400.0
HOME_OFFSET = 0.000000
HOME_SEARCH_VEL = -10.000000
HOME_LATCH_VEL = 0.500000
HOME_FINAL_VEL = 0.000000
HOME_USE_INDEX = NO

#********************
# Axis Y
#********************
[AXIS_1]
```

```
TYPE = LINEAR
HOME = 0.0
FERROR = 10.0
MIN_FERROR = 1.0
MAX_VELOCITY = 75.0
MAX_ACCELERATION = 750.0
# The values below should be 25% larger than MAX_VELOCITY and MAX_ACCELERATION
# If using BACKLASH compensation STEPGEN_MAXACCEL should be 100% larger.
#STEPGEN_MAXVEL = 31.25
STEPGEN_MAXVEL = 60
STEPGEN_MAXACCEL = 937.50
P = 1000.0
I = 0.0
D = 0.0
FF0 = 0.0
FF1 = 1.0
FF2 = 0.0
BIAS = 0.0
DEADBAND = 0.0
MAX_OUTPUT = 0.0
# these are in nanoseconds
DIRSETUP   = 14000
DIRHOLD    = 14000
STEPLEN    = 14000
STEPSPACE  = 14000
STEP_SCALE = 200.0
MIN_LIMIT = -0.01
MAX_LIMIT = 400.0
HOME_OFFSET = 0.000000
HOME_SEARCH_VEL = -10.000000
HOME_LATCH_VEL = 0.500000
HOME_FINAL_VEL = 0.000000
HOME_USE_INDEX = NO

#********************
# Axis Z
#********************
[AXIS_2]
TYPE = LINEAR
HOME = 0.0
FERROR = 10.0
MIN_FERROR = 1.0
MAX_VELOCITY = 25.0
MAX_ACCELERATION = 750.0
# The values below should be 25% larger than MAX_VELOCITY and MAX_ACCELERATION
# If using BACKLASH compensation STEPGEN_MAXACCEL should be 100% larger.
STEPGEN_MAXVEL = 31.25
STEPGEN_MAXACCEL = 937.50
```

```
P = 1000.0
I = 0.0
D = 0.0
FF0 = 0.0
FF1 = 1.0
FF2 = 0.0
BIAS = 0.0
DEADBAND = 0.0
MAX_OUTPUT = 0.0
# these are in nanoseconds
DIRSETUP   = 7000
DIRHOLD    = 7000
STEPLEN    = 7000
STEPSPACE  = 7000
STEP_SCALE = 200.0
MIN_LIMIT = -0.01
MAX_LIMIT = 400.0
HOME_OFFSET = 0.0

#********************
# Spindle
#********************
[SPINDLE_9]
P = 0
I = 0
D = 0
FF0 = 1
FF1 = 0
FF2 = 0
BIAS = 0
DEADBAND = 0
MAX_OUTPUT = 2000
```

Hal Configuration:

```
# Generated by PNCconf at Mon Nov 19 09:02:55 2018
# If you make changes to this file, they will be
# overwritten when you run PNCconf again

loadrt trivkins
loadrt [EMCMOT]EMCMOT servo_period_nsec=[EMCMOT]SERVO_PERIOD num_joints=[TRAJ]AXES
loadrt hostmot2
loadrt hm2_eth board_ip="192.168.1.121" config=" num_encoders=0 num_pwmgens=0
num_stepgens=3 sserial_port_0=0000xx"
setp    hm2_7i92.0.watchdog.timeout_ns 5000000
loadrt pid names=pid.x,pid.y,pid.z,pid.s

addf hm2_7i92.0.read        servo-thread
```

```
addf motion-command-handler   servo-thread
addf motion-controller        servo-thread
addf pid.x.do-pid-calcs       servo-thread
addf pid.y.do-pid-calcs       servo-thread
addf pid.z.do-pid-calcs       servo-thread
addf pid.s.do-pid-calcs       servo-thread
addf hm2_7i92.0.write         servo-thread
setp hm2_7i92.0.dpll.01.timer-us -50
setp hm2_7i92.0.stepgen.timer-number 1

# external output signals (see custom.hal)

# external input signals

# --- DIN-00 ---
net din-00    <=  hm2_7i92.0.gpio.014.in
net din-01    <=  hm2_7i92.0.gpio.009.in
net din-02    <=  hm2_7i92.0.gpio.010.in
net din-03    <=  hm2_7i92.0.gpio.011.in

# --- HOME-X ---
net home-x    <=  hm2_7i92.0.gpio.006.in_not

# --- HOME-Y ---
net home-y    <=  hm2_7i92.0.gpio.007.in_not


#********************
#  AXIS X
#********************

setp   pid.x.Pgain     [AXIS_0]P
setp   pid.x.Igain     [AXIS_0]I
setp   pid.x.Dgain     [AXIS_0]D
setp   pid.x.bias      [AXIS_0]BIAS
setp   pid.x.FF0       [AXIS_0]FF0
setp   pid.x.FF1       [AXIS_0]FF1
setp   pid.x.FF2       [AXIS_0]FF2
setp   pid.x.deadband  [AXIS_0]DEADBAND
setp   pid.x.maxoutput [AXIS_0]MAX_OUTPUT
setp   pid.x.error-previous-target true
setp   pid.x.maxerror .0005

net x-index-enable  <=> pid.x.index-enable
net x-enable        =>  pid.x.enable
net x-pos-cmd       =>  pid.x.command
net x-vel-cmd       =>  pid.x.command-deriv
net x-pos-fb        =>  pid.x.feedback
```

```
net x-output       =>  pid.x.output

# Step Gen signals/setup

setp   hm2_7i92.0.stepgen.00.dirsetup       [AXIS_0]DIRSETUP
setp   hm2_7i92.0.stepgen.00.dirhold        [AXIS_0]DIRHOLD
setp   hm2_7i92.0.stepgen.00.steplen        [AXIS_0]STEPLEN
setp   hm2_7i92.0.stepgen.00.stepspace      [AXIS_0]STEPSPACE
setp   hm2_7i92.0.stepgen.00.position-scale  [AXIS_0]STEP_SCALE
setp   hm2_7i92.0.stepgen.00.step_type       0
setp   hm2_7i92.0.stepgen.00.control-type    1
setp   hm2_7i92.0.stepgen.00.maxaccel        [AXIS_0]STEPGEN_MAXACCEL
setp   hm2_7i92.0.stepgen.00.maxvel          [AXIS_0]STEPGEN_MAXVEL

# ---closedloop stepper signals---

net x-pos-cmd    <= axis.0.motor-pos-cmd
net x-vel-cmd    <= axis.0.joint-vel-cmd
net x-output     => hm2_7i92.0.stepgen.00.velocity-cmd
net x-pos-fb     <= hm2_7i92.0.stepgen.00.position-fb
net x-pos-fb     => axis.0.motor-pos-fb
net x-enable     <= axis.0.amp-enable-out
net x-enable     => hm2_7i92.0.stepgen.00.enable

# ---setup home / limit switch signals---

net home-x     =>  axis.0.home-sw-in
net x-neg-limit     =>  axis.0.neg-lim-sw-in
net x-pos-limit     =>  axis.0.pos-lim-sw-in

#********************
#  AXIS Y
#********************

setp   pid.y.Pgain     [AXIS_1]P
setp   pid.y.Igain     [AXIS_1]I
setp   pid.y.Dgain     [AXIS_1]D
setp   pid.y.bias      [AXIS_1]BIAS
setp   pid.y.FF0       [AXIS_1]FF0
setp   pid.y.FF1       [AXIS_1]FF1
setp   pid.y.FF2       [AXIS_1]FF2
setp   pid.y.deadband  [AXIS_1]DEADBAND
setp   pid.y.maxoutput [AXIS_1]MAX_OUTPUT
setp   pid.y.error-previous-target true
setp   pid.y.maxerror .0005

net y-index-enable  <=> pid.y.index-enable
net y-enable       =>  pid.y.enable
```

```
net y-pos-cmd      =>  pid.y.command
net y-vel-cmd      =>  pid.y.command-deriv
net y-pos-fb       =>  pid.y.feedback
net y-output       =>  pid.y.output

# Step Gen signals/setup

setp   hm2_7i92.0.stepgen.01.dirsetup       [AXIS_1]DIRSETUP
setp   hm2_7i92.0.stepgen.01.dirhold        [AXIS_1]DIRHOLD
setp   hm2_7i92.0.stepgen.01.steplen        [AXIS_1]STEPLEN
setp   hm2_7i92.0.stepgen.01.stepspace      [AXIS_1]STEPSPACE
setp   hm2_7i92.0.stepgen.01.position-scale  [AXIS_1]STEP_SCALE
setp   hm2_7i92.0.stepgen.01.step_type       0
setp   hm2_7i92.0.stepgen.01.control-type    1
setp   hm2_7i92.0.stepgen.01.maxaccel        [AXIS_1]STEPGEN_MAXACCEL
setp   hm2_7i92.0.stepgen.01.maxvel          [AXIS_1]STEPGEN_MAXVEL

# ---closedloop stepper signals---

net y-pos-cmd    <= axis.1.motor-pos-cmd
net y-vel-cmd    <= axis.1.joint-vel-cmd
net y-output     => hm2_7i92.0.stepgen.01.velocity-cmd
net y-pos-fb     <= hm2_7i92.0.stepgen.01.position-fb
net y-pos-fb     => axis.1.motor-pos-fb
net y-enable     <= axis.1.amp-enable-out
net y-enable     => hm2_7i92.0.stepgen.01.enable

# ---setup home / limit switch signals---

net home-y     =>  axis.1.home-sw-in
net y-neg-limit     =>  axis.1.neg-lim-sw-in
net y-pos-limit     =>  axis.1.pos-lim-sw-in

#********************
#  AXIS Z
#********************

setp   pid.z.Pgain     [AXIS_2]P
setp   pid.z.Igain     [AXIS_2]I
setp   pid.z.Dgain     [AXIS_2]D
setp   pid.z.bias      [AXIS_2]BIAS
setp   pid.z.FF0       [AXIS_2]FF0
setp   pid.z.FF1       [AXIS_2]FF1
setp   pid.z.FF2       [AXIS_2]FF2
setp   pid.z.deadband  [AXIS_2]DEADBAND
setp   pid.z.maxoutput [AXIS_2]MAX_OUTPUT
setp   pid.z.error-previous-target true
setp   pid.z.maxerror .0005
```

```
net z-index-enable  <=> pid.z.index-enable
net z-enable       =>  pid.z.enable
net z-pos-cmd      =>  pid.z.command
net z-vel-cmd      =>  pid.z.command-deriv
net z-pos-fb       =>  pid.z.feedback
net z-output       =>  pid.z.output

# Step Gen signals/setup

setp   hm2_7i92.0.stepgen.02.dirsetup       [AXIS_2]DIRSETUP
setp   hm2_7i92.0.stepgen.02.dirhold        [AXIS_2]DIRHOLD
setp   hm2_7i92.0.stepgen.02.steplen        [AXIS_2]STEPLEN
setp   hm2_7i92.0.stepgen.02.stepspace      [AXIS_2]STEPSPACE
setp   hm2_7i92.0.stepgen.02.position-scale  [AXIS_2]STEP_SCALE
setp   hm2_7i92.0.stepgen.02.step_type       0
setp   hm2_7i92.0.stepgen.02.control-type    1
setp   hm2_7i92.0.stepgen.02.maxaccel        [AXIS_2]STEPGEN_MAXACCEL
setp   hm2_7i92.0.stepgen.02.maxvel          [AXIS_2]STEPGEN_MAXVEL

# ---closedloop stepper signals---

net z-pos-cmd    <= axis.2.motor-pos-cmd
net z-vel-cmd    <= axis.2.joint-vel-cmd
net z-output     => hm2_7i92.0.stepgen.02.velocity-cmd
net z-pos-fb     <= hm2_7i92.0.stepgen.02.position-fb
net z-pos-fb     => axis.2.motor-pos-fb
net z-enable     <= axis.2.amp-enable-out
net z-enable     => hm2_7i92.0.stepgen.02.enable

# ---setup home / limit switch signals---

net z-home-sw     =>  axis.2.home-sw-in
net z-neg-limit    =>  axis.2.neg-lim-sw-in
net z-pos-limit    =>  axis.2.pos-lim-sw-in

#*******************
#  SPINDLE S
#*******************

setp   pid.s.Pgain     [SPINDLE_9]P
setp   pid.s.Igain     [SPINDLE_9]I
setp   pid.s.Dgain     [SPINDLE_9]D
setp   pid.s.bias      [SPINDLE_9]BIAS
setp   pid.s.FF0       [SPINDLE_9]FF0
setp   pid.s.FF1       [SPINDLE_9]FF1
setp   pid.s.FF2       [SPINDLE_9]FF2
setp   pid.s.deadband  [SPINDLE_9]DEADBAND
```

```
setp   pid.s.maxoutput [SPINDLE_9]MAX_OUTPUT
setp   pid.s.error-previous-target true

net spindle-index-enable  <=> pid.s.index-enable
net spindle-enable       =>  pid.s.enable
net spindle-vel-cmd-rpm     => pid.s.command
net spindle-vel-fb-rpm      => pid.s.feedback
net spindle-output       <=  pid.s.output

# ---setup spindle control signals---

net spindle-vel-cmd-rps       <=  motion.spindle-speed-out-rps
net spindle-vel-cmd-rps-abs   <=  motion.spindle-speed-out-rps-abs
net spindle-vel-cmd-rpm       <=  motion.spindle-speed-out
net spindle-vel-cmd-rpm-abs   <=  motion.spindle-speed-out-abs
net spindle-enable            <=  motion.spindle-on
net spindle-cw               <=  motion.spindle-forward
net spindle-ccw               <=  motion.spindle-reverse
net spindle-brake             <=  motion.spindle-brake
net spindle-revs              =>  motion.spindle-revs
net spindle-at-speed          =>  motion.spindle-at-speed
net spindle-vel-fb-rps        =>  motion.spindle-speed-in
net spindle-index-enable     <=>  motion.spindle-index-enable

# ---Setup spindle at speed signals---

sets spindle-at-speed true


#*****************************
# connect miscellaneous signals
#*****************************

#  ---HALUI signals---

net joint-select-a       halui.joint.0.select
net x-is-homed           halui.joint.0.is-homed
net jog-x-pos           halui.jog.0.plus
net jog-x-neg           halui.jog.0.minus
net jog-x-analog         halui.jog.0.analog
net joint-select-b       halui.joint.1.select
net y-is-homed           halui.joint.1.is-homed
net jog-y-pos           halui.jog.1.plus
net jog-y-neg           halui.jog.1.minus
net jog-y-analog         halui.jog.1.analog
net joint-select-c       halui.joint.2.select
net z-is-homed           halui.joint.2.is-homed
net jog-z-pos           halui.jog.2.plus
```

```
net jog-z-neg            halui.jog.2.minus
net jog-z-analog         halui.jog.2.analog
net jog-selected-pos     halui.jog.selected.plus
net jog-selected-neg     halui.jog.selected.minus
net spindle-manual-cw    halui.spindle.forward
net spindle-manual-ccw   halui.spindle.reverse
net spindle-manual-stop  halui.spindle.stop
net machine-is-on        halui.machine.is-on
net jog-speed            halui.jog-speed
net MDI-mode             halui.mode.is-mdi

#  ---coolant signals---

net coolant-mist     <=  iocontrol.0.coolant-mist
net coolant-flood    <=  iocontrol.0.coolant-flood

#  ---probe signal---

net probe-in     =>  motion.probe-input

#  ---motion control signals---

net in-position             <=  motion.in-position
net machine-is-enabled      <=  motion.motion-enabled

#  ---digital in / out signals---

net din-00     =>  motion.digital-in-00
net din-01     =>  motion.digital-in-01
net din-02     =>  motion.digital-in-02
net din-03     =>  motion.digital-in-03

#  ---estop signals---

net estop-out     <=  iocontrol.0.user-enable-out
net estop-out     =>  iocontrol.0.emc-enable-in

#  ---toolchange signals for custom tool changer---

net tool-number           <=  iocontrol.0.tool-prep-number
net tool-change-request   <=  iocontrol.0.tool-change
net tool-change-confirmed  =>  iocontrol.0.tool-changed
net tool-prepare-request   <=  iocontrol.0.tool-prepare
net tool-prepare-confirmed  =>  iocontrol.0.tool-prepared
```

Custom Hal Configuration:

```
# Include your custom HAL commands here
#
This file will not be overwritten when you run PNCconf again

#

loadrt

flipflop count=2
addf
flipflop.0 servo-thread
addf
flipflop.1 servo-thread

setp

hm2_7i92.0.gpio.012.is_output true
setp
hm2_7i92.0.gpio.013.is_output true

net dout-0-on

motion.digital-out-00 flipflop.0.set
net dout-0-off
motion.digital-out-01 flipflop.0.reset

net dout-1-on
motion.digital-out-02 flipflop.1.set
net dout-1-off
motion.digital-out-03 flipflop.1.reset

net dout-00

flipflop.0.out hm2_7i92.0.gpio.012.out
net dout-01
flipflop.1.out hm2_7i92.0.gpio.013.out
```