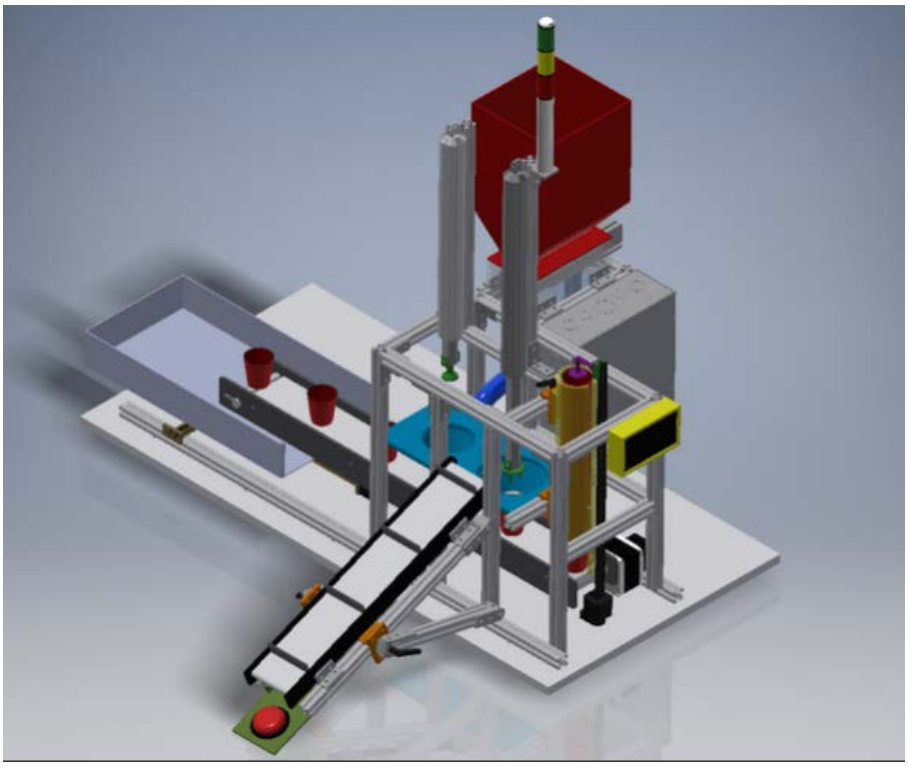
Assisted Packaging Project for Johnson County Developmental Supports



Summary Report

Rehabilitation Engineering Advancement in Kansas (BREAK) program
Team Members: Brian Gier, Sean McBride, Kyle Young
Sponsor: Dr. Kenneth Fischer
Advisor: Thomas DeAgostino
Funded by: National Science Foundation
Beneficiary: Johnson County Developmental Supports

JOHNSON COUNTY KANSAS
Developmental Supports

Project

This project is a partnership between The University of Kansas, Department of Mechanical Engineering and Johnson County Developmental Supports (JCDS) in Lenexa, KS. JCDS specializes in employing persons with physical and mental disabilities that limit employment opportunities elsewhere in the community. Employees' wages are based on a per-part production output, therefore maximum output of components is ideal for their success. In an effort to increase wages, completion rates, and employment desirability at other organizations, JCDS has requested a device to be designed that will improve these aspects for its employees. The end-product assembly consists of three components: a urine sample cup, a syringe hub, and two vinyl gloves.

The design process for this project began with meeting with Mr. Brian Skibbe of JCDS and observing the work environment for the intended user. The device specifications that were requested structured the design process for this project.

Device Description

The frame for the device is constructed using T-slotted Aluminum (80/20), sliders, and screws. A Mini Mover conveyor belt that is centrally located in the machine moves the cups to each station. A cleated Mini Mover conveyor belt extends out of the machine so that the seated user can place pairs of vinyl gloves in the spaces between the cleats. A clear plastic tube that holds a stack of cups is mounted on the frame that lines up with a Haydon Kerk linear rail actuator. Attached to the rail actuator is an extended arm that is used to push cups onto the central conveyor. The third component of the device consists of a Delrin cylinder, a stepper motor, a funnel and a clear flexible tube. A hole approximately the size of one hub has been drilled in the Delrin cylinder as to allow only one hub to enter the packaging process. Attached to the cylinder is a stepper motor which turns exactly one rotation. Directly above the cylinder is the vertical, flexible tube which acts as a feeder into the cylinder and directly above the tube is a funnel where hubs can be poured in.

Above the central conveyor belt are two linear actuators and two funnels. Each funnel has been specifically designed for the glove and hub packaging processes. Above each respective funnel is a linear actuator to stuff each of the components into the plastic cup; first the gloves are slightly stuffed into the cup, next the hub is stuffed and compactly packages the entire product. Attached to the bottom of each linear actuator are specifically designed aluminum pieces that effectively stuff each of their respective components.

A 12V power supply is connected to two terminal strips that share the input voltage with each component. One 3-pronged plug must be plugged into a normal 110/120V wall

outlet. This runs to a switch that passes the input voltage to four 3-pronged outlets mounted on the machine. The two Mini Mover conveyor belts and the 12V power supply have 3-pronged plugs and the Arduino power cord has two prongs that all need to be plugged into these outlets. The attached wiring diagram shows how each component is wired to both power and ground. The wires for the buttons must be soldered onto specific pins on the female 3.5mm jack and the wires for the solenoid actuator are specific as well. These are shown in the wiring diagram.

Programming

The machine is controlled by an Arduino Mega 2560. This is a microcontroller that runs programs in the C++ programming language to control inputs and outputs. The program sets two input pins for the two buttons, one to advance the machine and one to reload the cups if desired before 26 cups have been dropped from the tube. The program sets 16 outputs to turn on various relays or to send power directly to the components. For the Haydon Kerk linear rail actuator, there is a programmable microcontroller that uses a proprietary programming language. It runs independently of the Arduino code, but can utilize inputs and outputs to react to the Arduino. Documentation on using the Haydon Kerk Idea Drive software is included in the binder and on the flashdrive.

The code defines the Arduino digital outputs 22-50 to send signals to relays 1-14 and to the rotary motor. It uses the Stepper Library to define running variables for the motor. Digital pins 2 through 4 are set as inputs for buttons. Functions are declared for each process (moving the main belt, the cleated belt, the linear actuators, the roller and solenoid, and outputs for the rail actuator code. The main loop for the program is a progressive loop that has to work through an initialization step before a unit can be produced. The process involves five different stages, before the fifth is repeated until materials expire (Fig. 1).

Due to a signal delay between the Arduino and the Haydon Kerk programmable microcontroller, the code tells the rail to move at the end of each step. The code currently runs each step incrementally, but a Scheduler could be added to make multiple components move simultaneously. When the cup arm has moved down to the bottom of the cutout on the cup tube, the carriage automatically returns to the top to allow for the cups to be refilled and to continue where the machine left off. The rail actuator program utilizes subroutines that are triggered by inputs from the Arduino (Fig. 2). If Relay 13 is turned on, Input 1 in the program will be set to LOW and the cup arm will move to the bottom face of the first cup. If Relay 12 is turned on, Input 2 tells the cup arm to move less than half an inch, dropping only one cup onto the conveyor. Input 3 or 4 can be set to have an interrupt that allows the cup tube to be refilled at any time.

Operation

The machine is designed in such a way that one sitting person can primarily operate it

and one standing person can refill materials periodically and infrequently. At the beginning of the day, the switch on the back should be switched on, which is the downward position. Upon powering on, the rail actuator with the plunger that dispenses cups will automatically move to the top of the rail. The locking arm on the slider for the cup tube should be swung counterclockwise and the tube should be moved so that cups can be placed into the tube. The machine is designed to have 29 cups loaded into it at the beginning of operation. Hubs can be dumped into the funnel and lined up in the tube. Gloves should be placed above the cleat closest to the operator. The button can now be pressed. Placing gloves and pressing the button should be repeated until materials run out.

The first time the button is pressed, the cleated conveyor moves the first pair of gloves moves one step and the cup plunger moves down to the first cup in the tube. The second time, the cleated conveyor moves one more step and the plunger pushes the first cup out of the tube and onto the main conveyor belt. After the third press, the cleated conveyor moves another step, the main belt moves forward, and the plunger drops another cup out of the tube. The fourth press causes the cleated conveyor to move one more step, dropping the first pair of gloves into the blue funnel above the first cup. The first linear actuator plunges three times, pushing the gloves into the cup. The main conveyor moves one step, arranging the first cup under the second funnel and the second cup under the first funnel. The roller turns one full revolution and drops a hub down the chute and into the blue funnel. A solenoid actuator extends to hold the hubs in the tube so that they do not jam the machine. The plunger drops the third cup onto the main conveyor. After the fifth and every subsequent press of the button, the cleated conveyor moves one step, dropping a pair of gloves into the blue funnel. Both linear actuators plunge to push gloves and a hub, respectively, into the cups. The main conveyor moves forward one step, the roller drops a hub, and the solenoid lets a hub fall into the roller. The cup plunger drops a cup, until it reaches the bottom of the cutout in the tube.

Once the cup plunger on the linear rail actuator reaches the bottom of the tube, it will automatically move back to the top so the cups can be refilled.

1. Turn switch on
2. Wait for cup plunger to reach top
3. Slide cup tube out and refill with 29 cups (at brim when pushed down)
4. Fill hub tube and funnel
5. Place two gloves in first spot closest to the user
6. Push button
7. Place two gloves in next available spot closest to the user
8. Repeat 5-6 until cups, hubs, or gloves need to be refilled

Future Work

- Improve hub feeder system
    - Prevent jams in funnel and roller
    - Implement a hopper
    - Use better material/3D print new roller
- Cup tipping over from gloves under funnel
    - Grind out more material from funnel
    - Rearrange components
    - Reprint a new shape
- Kan-Ban light
    - Implement for interrupts or materials running low
    - Use small LEDs that are magnified
    - Use existing, that requires 24V so use transistors
- Display screen
    - Count completed units
    - Make button to reset machine
- Output
- Protective Casing
    - Plexiglas
    - Plastic
- Sensors
    - Verify materials get to proper position
- Improve mounting for the rotary motor
    - Protect wires
    - Reduce pinch points
- Mount wiring box
- Move plugs under wiring box
- Include stopper to position cup tube to be centered under cup arm
- Improve glove stuffer
    - Spread out rods to cover more area

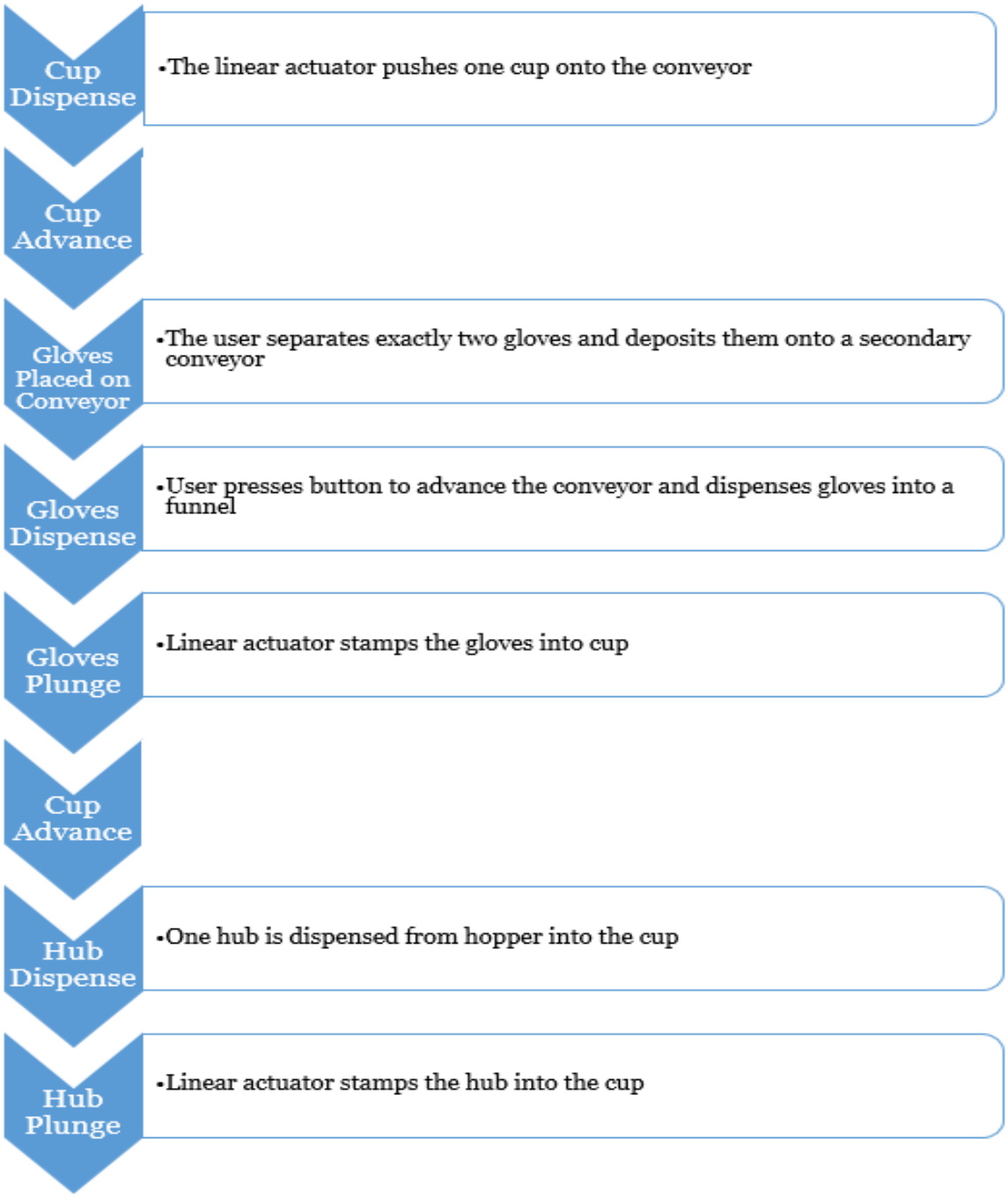


Fig. 1: Flow Chart

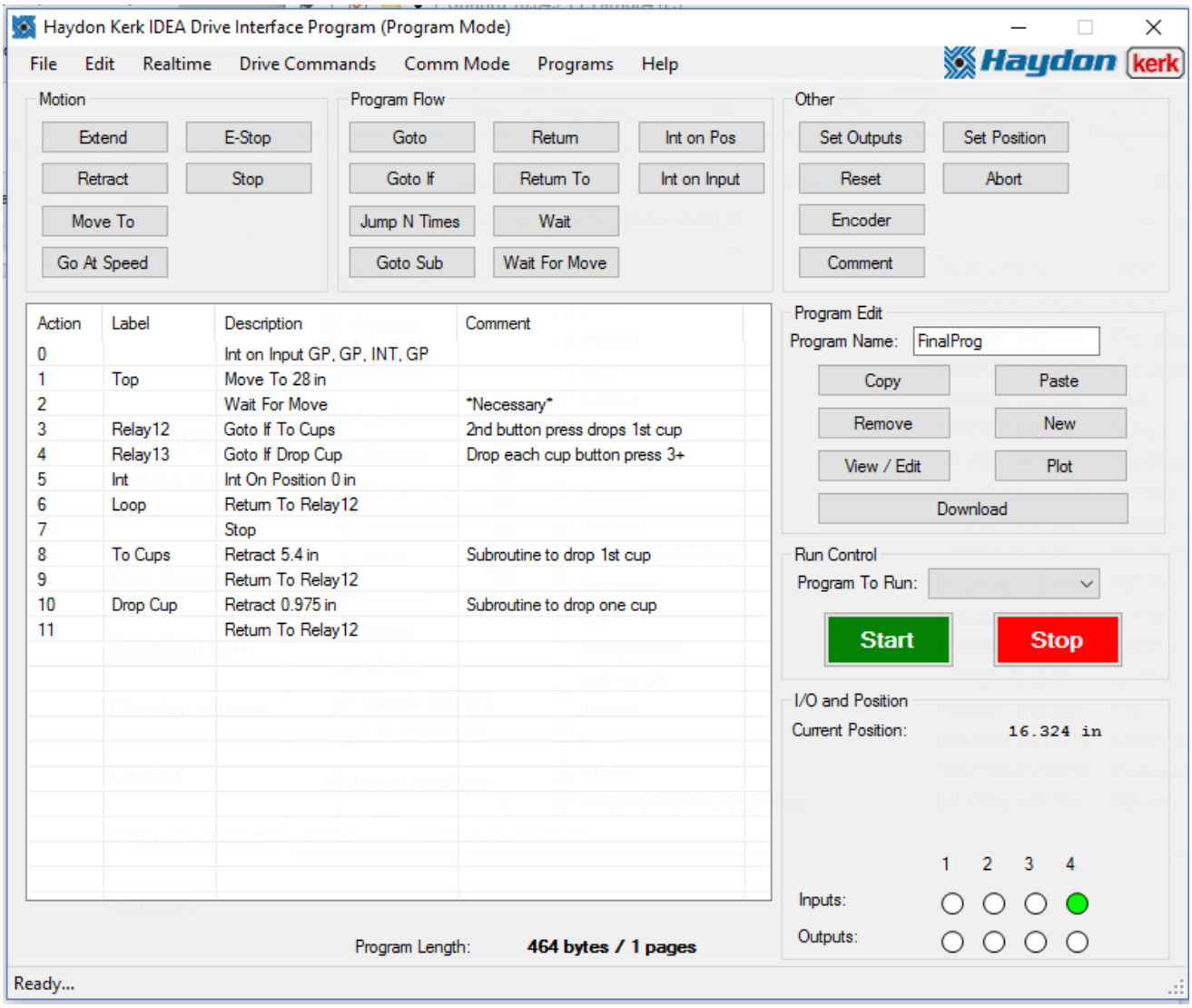


Fig. 2: Haydon Kerk Rail Actuator Code

```
#include <Arduino.h> //General Arduino Library
#include <Stepper.h> //Library for the rotary motor
const int stepsPerRevolution = 6400; //Number of steps unique to motor
Stepper myStepper(stepsPerRevolution, 48, 49); //Defining Arduino pins

//#define LOW 0 //These variables stopped working
//#define HIGH 1
//Buttons:Pin locations for buttons on Mega
#define Fwd_Button  2 //Blue button that is used in normal process
#define Cups_Button 3 //Moves rail actuator to refill cups
#define EStop 4 //Cancels process

//Stuffer1
#define Relay_5  30     // stuffer fwd
#define Relay_6  32     // stuffer fwd
#define Relay_7  34     // stuffer reverse
#define Relay_8  36     // stuffer reverse

//Stuffer2
#define Relay_1  22     // stuffer fwd
#define Relay_2  24     // stuffer fwd
#define Relay_3  26     // stuffer reverse
#define Relay_4  28     // stuffer reverse

//Belt_Main: Fwd only, green wire for reverse
#define Relay_10 40  // belt going fwd

//Belt_Gloves: Fwd only, green wire for reverse
#define Relay_9 38

//Hub Roller
#define Direction 49
#define clock2 48
int StepCounter = 0;
#define Relay_11 42 //Solenoid to hold 2nd to last hub

//Rail
#define Relay_12 44 //Input 2 in Haydon Kerk Idea Drive program
#define Relay_13 46 //Input 1 in Haydon Kerk Idea Drive program
#define Relay_14 51 //Input 3 in Haydon Kerk Idea Drive program

int count = 1; //Keep track of completed units, for display screen integration
int trial = 0;

void setup() {
  // put your setup code here, to run once:
```

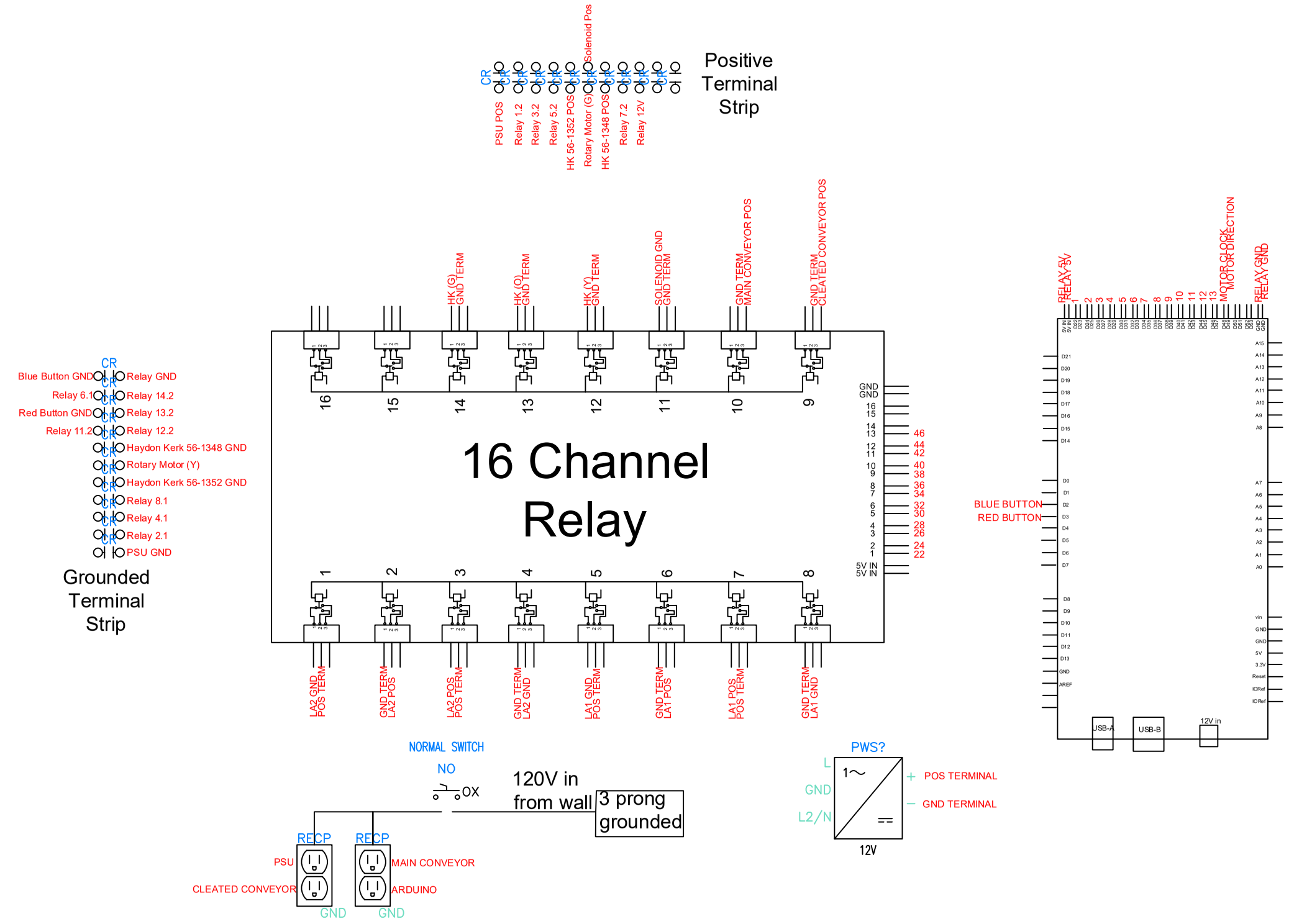
```
 // === Initialize Pins and Reset ===
 digitalWrite(Relay_1, HIGH);
 digitalWrite(Relay_2, HIGH);
 digitalWrite(Relay_3, HIGH);
 digitalWrite(Relay_4, HIGH);
 digitalWrite(Relay_5, HIGH);
 digitalWrite(Relay_6, HIGH);
 digitalWrite(Relay_7, HIGH);
 digitalWrite(Relay_8, HIGH);
 digitalWrite(Relay_9, HIGH);
 digitalWrite(Relay_10, HIGH);
 digitalWrite(Relay_11, HIGH);
 digitalWrite(Relay_12, HIGH);
 digitalWrite(Relay_13, HIGH);
 digitalWrite(Relay_14, HIGH);
 myStepper.setSpeed(60); //Roller speed
 Serial.begin(9600);

 // === Set pins as Outputs ===
 pinMode(Relay_1, OUTPUT);
 pinMode(Relay_2, OUTPUT);
 pinMode(Relay_3, OUTPUT);
 pinMode(Relay_4, OUTPUT);
 pinMode(Relay_5, OUTPUT);
 pinMode(Relay_6, OUTPUT);
 pinMode(Relay_7, OUTPUT);
 pinMode(Relay_8, OUTPUT);
 pinMode(Relay_9, OUTPUT);
 pinMode(Relay_10, OUTPUT);
 pinMode(Relay_11, OUTPUT);
 pinMode(Relay_12, OUTPUT);
 pinMode(Relay_13, OUTPUT);
 pinMode(Relay_14, OUTPUT);
 delay(3000);

 //== Set pins as Inputs==
 pinMode(Fwd_Button, INPUT_PULLUP);
 pinMode(Cups_Button, INPUT_PULLUP);
 pinMode(EStop, INPUT_PULLUP);
 Serial.begin(9600);
}

//Main conveyor, timing matches distance between funnels
void BeltFwd() {
 digitalWrite(Relay_10,LOW);
 delay(460);
```

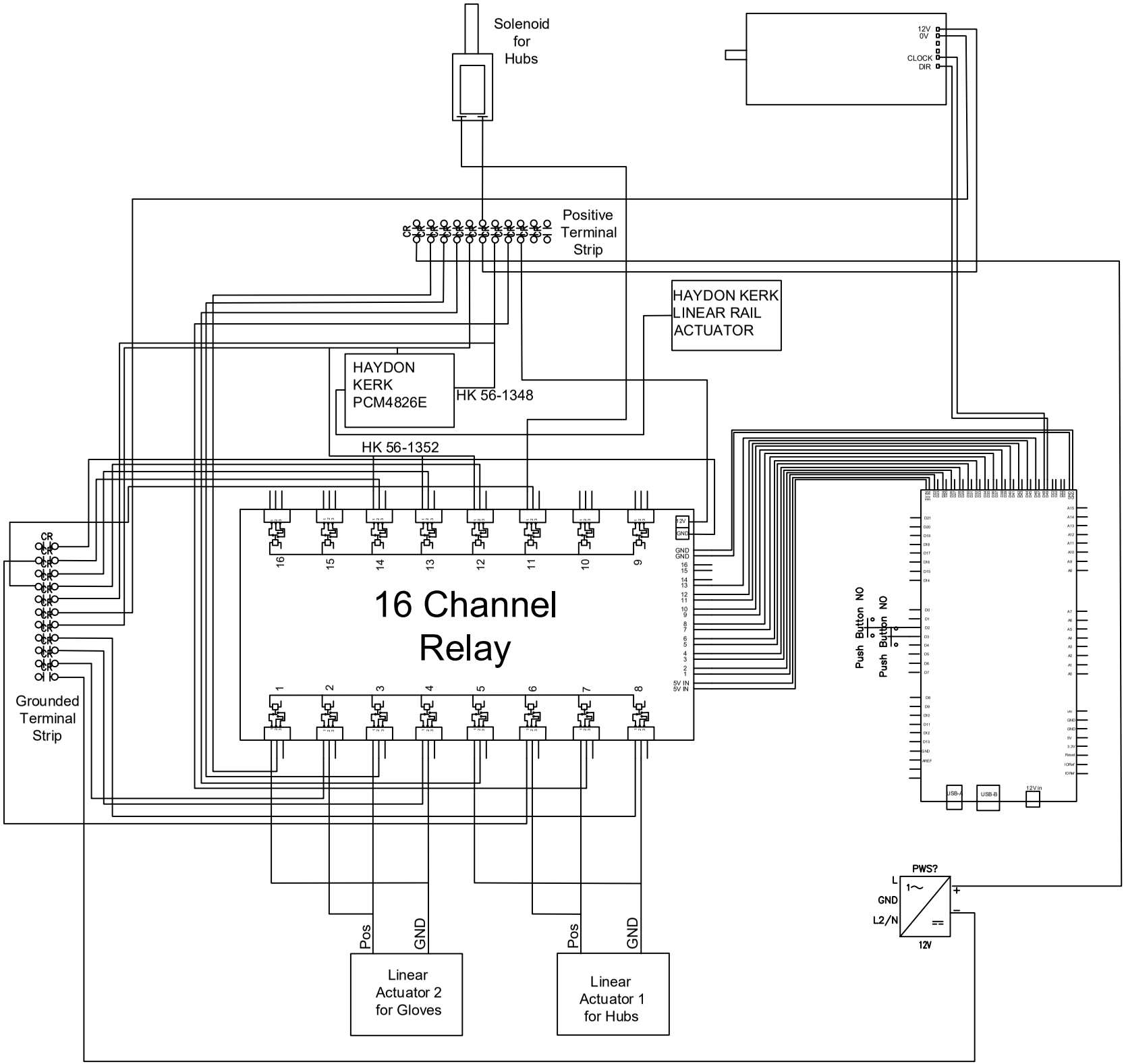
```
 digitalWrite(Relay_10, HIGH);
 delay(500);
} // End of BeltFwd

//Cleated conveyor, not exact yet
void CleatsFwd() {
 digitalWrite(Relay_9,LOW);
 delay(2870);
 digitalWrite(Relay_9, HIGH);
 delay(500);
} // End of BeltFwd

void Stuffer1() {
 //---( stuffer going fwd)---
 digitalWrite(Relay_5,LOW);
 digitalWrite(Relay_6,LOW);
 delay(1900);

 //---( stuffer fwd relay off)---
 digitalWrite(Relay_5, HIGH);
 digitalWrite(Relay_6, HIGH);
 delay(500);

 //---( stuffer going back to rest position)---
 digitalWrite(Relay_7,LOW);
 digitalWrite(Relay_8,LOW);
 delay(700);  //wait for a second

 //---( stuffer reverse relays off)---
 digitalWrite(Relay_7, HIGH);
 digitalWrite(Relay_8, HIGH);
 delay(100);

 digitalWrite(Relay_5,LOW);
 digitalWrite(Relay_6,LOW);
 delay(700);

 //---( stuffer fwd relay off)---
 digitalWrite(Relay_5, HIGH);
 digitalWrite(Relay_6, HIGH);
 delay(250);

 //---( stuffer going back to rest position)---
 digitalWrite(Relay_7,LOW);
 digitalWrite(Relay_8,LOW);
 delay(700);  //wait for a second
```

```
  //---( stuffer reverse relays off)---
  digitalWrite(Relay_7, HIGH);
  digitalWrite(Relay_8, HIGH);
  delay(100);

 digitalWrite(Relay_5,LOW);
 digitalWrite(Relay_6,LOW);
  delay(700);

  //---( stuffer fwd relay off)---
  digitalWrite(Relay_5, HIGH);
  digitalWrite(Relay_6, HIGH);
  delay(250);

  //---( stuffer going back to rest position)---
  digitalWrite(Relay_7, LOW);
  digitalWrite(Relay_8, LOW);
  delay(1900);

  //---( stuffer reverse relays off)---
  digitalWrite(Relay_7, HIGH);
  digitalWrite(Relay_8, HIGH);
  delay(100);
} // End of Stuffer

void Stuffer2() {
  //---( stuffer going fwd)---
  digitalWrite(Relay_1, LOW);
  digitalWrite(Relay_2, LOW);
  delay(950);
  //---( stuffer fwd relay off)---
  digitalWrite(Relay_1, HIGH);
  digitalWrite(Relay_2, HIGH);
  delay(500);
  //---( stuffer going back to rest position)---
  digitalWrite(Relay_3, LOW);
  digitalWrite(Relay_4, LOW);
   delay(950);  //wait for a second
  //---( stuffer reverse relays off)---
  digitalWrite(Relay_3, HIGH);
  digitalWrite(Relay_4, HIGH);
  delay(100);
} // End of Stuffer
```

```
//One hub falls into hole, solenoid holds other hubs up, roller turns, solenoid
drops a hub
void Roller(){
 myStepper.step(3200);
    delay(200);
 myStepper.step(-3200);
    delay(100);
 digitalWrite(Relay_11, HIGH);
 delay(200);
 digitalWrite(Relay_11, LOW);
  }

//Input 2 in Haydon Kerk Idea Drive program
void Rail1() {
 digitalWrite(Relay_12, LOW);
 delay(1000);
 digitalWrite(Relay_12, HIGH);
}

/*Input 1 in Haydon Kerk Idea Drive program
Moves cup arm to first cup, may drop a cup and needs to be accounted for*/
void Rail() {
 digitalWrite(Relay_13, LOW);
 delay(1000);
 digitalWrite(Relay_13, HIGH);
}

void Cups() {
 digitalWrite(Relay_14, LOW);
 delay(1000);
 digitalWrite(Relay_14, HIGH);
}


void loop() {
/*This is for IR sensors to track material inputs. Examples in the Sensors folder.
  //int sensorVal = analogRead(A0);
  //int sensorVal2 = analogRead(A1);
  //int sensorVal3 = analogRead(A2);
  //  Serial.println(sensorVal);
  //  Serial.println(sensorVal2);
  //   delay(1);
  //    if(sensorVal>=400){
  //      delay(100);
*/
if (digitalRead(Cups_Button) == LOW){
```

```
  Cups();
}
  else if (digitalRead(Fwd_Button) == LOW) {

    if (count == 1) {
      CleatsFwd(); //add hub control to this stage, add sensor to stop
      count++;
      delay(50);
    }
    else if (count == 2) {
      CleatsFwd();
      Rail1();
      count++;
      delay(50);
    }
    else if (count == 3) {
      CleatsFwd();
      BeltFwd();
      Rail();
      count++;
      delay(50);
    }
    else if (count == 4) {
      CleatsFwd();
      Stuffer1();
      BeltFwd();
      Roller();
      Rail();
      count++;
      delay(50);
    }
    else if (count >= 5) {
      CleatsFwd();
      Stuffer1();
      Stuffer2();
      BeltFwd();
      Roller();
      Rail();
      count++;
      delay(50);
    }    }    }

  //  cout count == 0; //For display screen integration
  //  trial ++;
```


Positive Terminal Strip
PSU POS
Relay 1.2
Relay 3.2
Relay 5.2
HK 56-1352 POS
Rotary Motor (G)
HK 56-1348 POS
Relay 7.2
Relay 12V
Solenoid Pos
Grounded Terminal Strip
Blue Button GND
Relay GND
Relay 6.1
Relay 14.2
Red Button GND
Relay 13.2
Relay 11.2
Relay 12.2
Haydon Kerk 56-1348 GND
Rotary Motor (Y)
Haydon Kerk 56-1352 GND
Relay 8.1
Relay 4.1
Relay 2.1
PSU GND
16 Channel Relay
HK (G) GND TERM
HK (O) GND TERM
HK (Y) GND TERM
SOLENOID GND GND TERM
GND TERM MAIN CONVEYOR POS
GND TERM CLEATED CONVEYOR POS
LA2 GND POS TERM
GND TERM LA2 POS
LA2 POS POS TERM
GND TERM LA2 GND
LA1 GND POS TERM
GND TERM LA1 POS
LA1 POS POS TERM
GND TERM LA1 GND
5V IN
RELAY 5V
RELAY GND
MOTOR CLOCK
MOTOR DIRECTION
BLUE BUTTON
RED BUTTON
USB-A
USB-B
12V in
NORMAL SWITCH
NO
120V in from wall
3 prong grounded
RECP
PSU
CLEATED CONVEYOR
MAIN CONVEYOR
ARDUINO
GND
PWS?
L
GND
L2/N
12V
POS TERMINAL
GND TERMINAL


Solenoid for Hubs
12V
0V
CLOCK
DIR
Positive Terminal Strip
HAYDON KERK LINEAR RAIL ACTUATOR
HAYDON KERK PCM4826E
HK 56-1348
HK 56-1352
16 Channel Relay
Grounded Terminal Strip
Push Button NO
Push Button NO
PWS?
L
GND
L2/N
12V
Pos
GND
Linear Actuator 2 for Gloves
Pos
GND
Linear Actuator 1 for Hubs