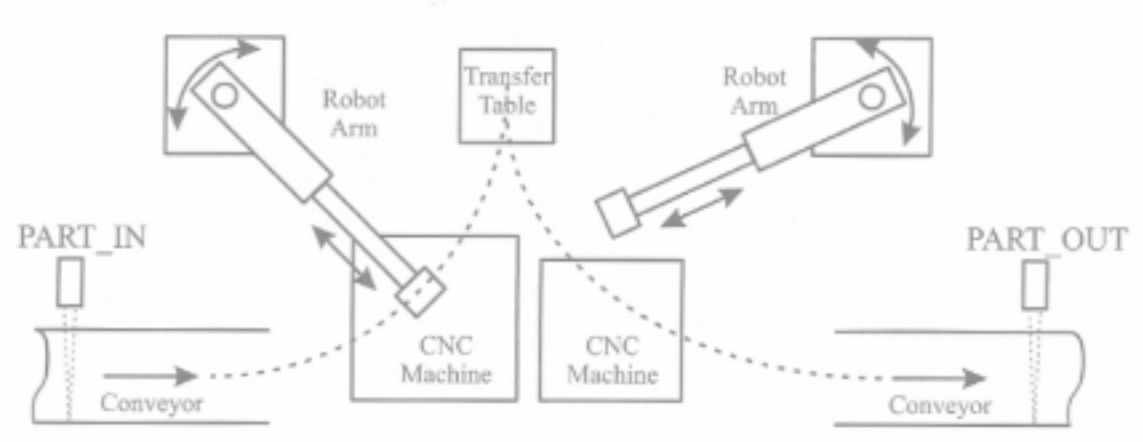
Automated Parts Transfer System



By Kurt Foster, 2017

Demonstration YouTube Link: https://youtu.be/VDCZo-pmgzg

Project Accomplishments:

- Two wafer handling robot arms (donated by HP Inc.) were integrated into a production line scenario utilizing two Arduino's and a PLC in a single project.

- I was the first mechatronics student to integrate the undocumented/unused robots into a functional system, programmed the Arduino's and PLC, and wired the system.

- Created full documentation, wiring schematic, ladder diagram, and Arduino code.

- My Mechatronics program instructor is now using this documentation to create a lab integrating a PLC and Arduino for future students.

This project was an assignment in my Advanced PLC Troubleshooting and Wiring class I took during the Fall term at LBCC in 2017. It was possible to combine assignments if it was demonstrated that the automated process created went well above and beyond the assumptions of the original system design. I incorporated a full terms' worth of labs into this one lab. In this document, I will compare what was given to me and how I went about completing the process.

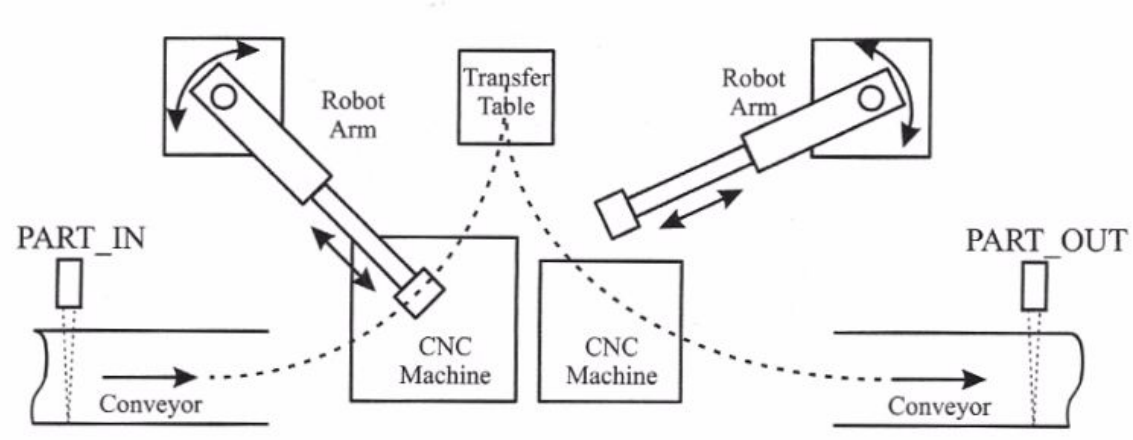


Figure 1: Problem Setup from Description

Problem Description: Parts move into and out of machining work cell via conveyors. One robot moves the unfinished part from the incoming conveyor to a CNC lathe which turns the rectangular part into an axle. This process takes 1 minute and then the robot moves the part to a transfer table when the second robot picks it up and moves it to a second CNC machine which measures, deburrs and polishes the axle, a process that takes 1 minute, 15 seconds. The PART_IN sensor input is momentarily triggered when a part enters the workcell and PART_OUT is triggered when a part leaves the work cell. A reset should be included in order to start the process over.

The following pages document what I decided to do for completion of the project.

Sections

Section 1: Description of the problem and process

Summary: The problem is to go from raw material to a finished part that successfully goes through two separate CNC machines. There are two main parts to the process, one that controls the robots and the other that keeps track of parts coming in and out.

Robot Control Process: A part goes on a moving conveyor and passes a sensor. The sensor will turn the conveyor off and then run the first robot arm sequence. The first robot arm starts at the transfer table unextended. It will go to the conveyor and then extend and then retract which picks the part up. First robot then moves to the CNC machine and then extends and retracts within about a second, dropping the part off at the first CNC machine. After a single minute, the robot arm extends and picks up the part and then retracts. It will then move the part to the transfer table. If more parts show up on the first conveyor, the whole first robot sequence restarts. Once a part has reached the transfer table, a sensor will get activated, and stay activated as long as there are parts sitting there, kind of like a vertical trey. As long as the sensor is activated, the second robot sequence will repeat itself. The second robot starts at the transfer table unextended. When the transfer table sensor is activated, the second robot arm extends to pick the part up and retracts over the course of a second. Robot two then rotates to the second CNC machine extends and retracts, dropping the part off at the CNC machine. The second CNC machine will run for one minute and fifteen seconds. After this time period, the robot arm will extend and retract to pick up the finished part from the CNC machine. Once the part is taken, robot two will rotate to the second conveyor and drop part off. After the part is dropped off the second conveyor will run. Five seconds after the second conveyor sensor is activated, the second conveyor will stop.

Part Tracking Process: Each time a part goes past the conveyor in sensor, it starts the process of the picking up and placing action. Each time this sensor is activated, it counts up a total parts counter within the PLC. If this total parts counter is equal to twenty-five parts, then the first section of the pick and place will stop until the total parts counter goes down. An exit sensor is placed at the conveyor out conveyor. Each time this sensor is activated, the total parts counter will go down as many times as the sensor parts out is activated. In between these sensors lies a part present sensor. This gets activated any time there is a part to be sent on to the second CNC machine of the whole process. When this sensor is active, the second part of the operation will always continue.

Section 2: Operation with Cylinders and Stepper Motors

Since the design of this system was not set in stone, I went off and put together a list of components required to power pneumatic cylinders for the linear action, which would be mounted on top of stepper motors. This makes use of electrically actuated directional control valves and limit switches for the cylinders.

Ultimately, I would use the internal stepper of each robot arm to control the linear action, which removed the need for pneumatic cylinders.

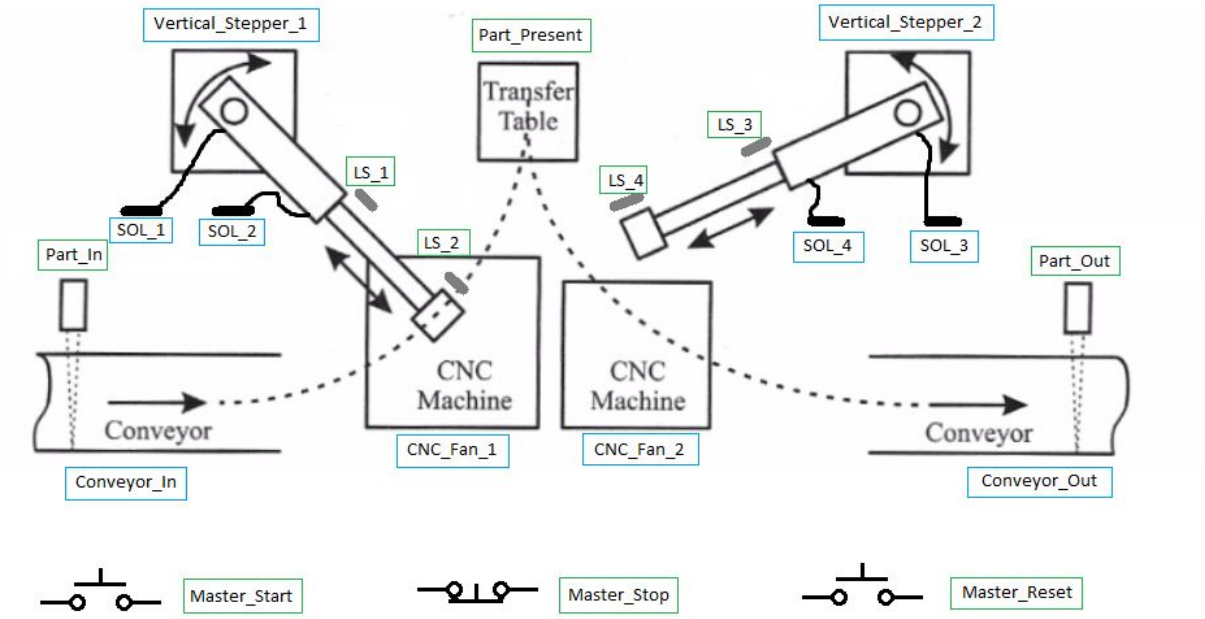


Figure 2: Possible system setup with limit switches and pneumatic cylinders attached to stepper motors. Blue square items are outputs, green square items are inputs

OPERATION AT A GLANCE:

- When the arm of the robot is extended the robot picks up the part.
- Turning on the CNC Machine will trigger the machining process.
- When the robot holds the part over the CNC Machine the part is in the CNC Machine.
- When a signal is sent to the Arduino to position the robot the robot is in position after a period of time.

- When a part is taken to the Transfer Table it is held and put in place for the second robot to pick up.
- Parts are placed on the Conveyor_In externally.
- Parts are removed from the Conveyor_Out externally.

START/STOP AND COUNTING:

- A Master_Start button will turn on Conveyor_In motor
- A Master_Stop button will cause all motors (and pneumatic cylinders if we use them) to come to a stop but keeps counters in memory (this is a easy integration, just put a relay contact at every "moving" output rung, it would have the same relay contact as the main start-stop memory latch rung)
- A Master_Reset button will reset all counters and robot arms to their normal positions, and it will cut power to both conveyors. (to start process again, click start) NOTE: Master_Stop must be pressed before reset can be pressed
- Part_In sensor will count up a Total_Counter
- Part_Out sensor will subtract (count down) a Total_Counter
- When Total_Counter reaches 25, Cell_25P indicator is turned on
- When Total_Counter reaches 25, Conveyor_In will turn off
- When Total_Counter reaches 25, it will latch a memory on a Count_25 internal bit
- When Total_Counter is less than or equal to 5 and Count_25 is latched, it will turn on Conveyor_In motor
- When Total_Counter is 5 or less (less than or equal to) and Total_Counter at 25 memory is latched, then Conveyor_In will turn on
- When Total_Counter is zero, the Cell_Empty indicator will flash

ROBOT ARM SEQUENCES GO AS FOLLOWS:

Robot Arm Starting Points: I will assume the starting positions of the robot arms are Unextended, and located at the Transfer Table at the start, this is position 3 and 4.

NOTE: Pneumatic cylinders are used for the extended and retracts, Robot_1 retracted will be powered by Sol_2, extended will be powered by Sol_1 … Robot_2 retracted will be powered by Sol_4, extended powered by Sol_3.

IF we use limit switches to determine the positions of the cylinders, the retracted Robot_Arm_1 cylinder will activate LS_1, extended will activate LS_2 … Robot_Arm_2 retracted will activate LS_3, extended will activate LS_4.

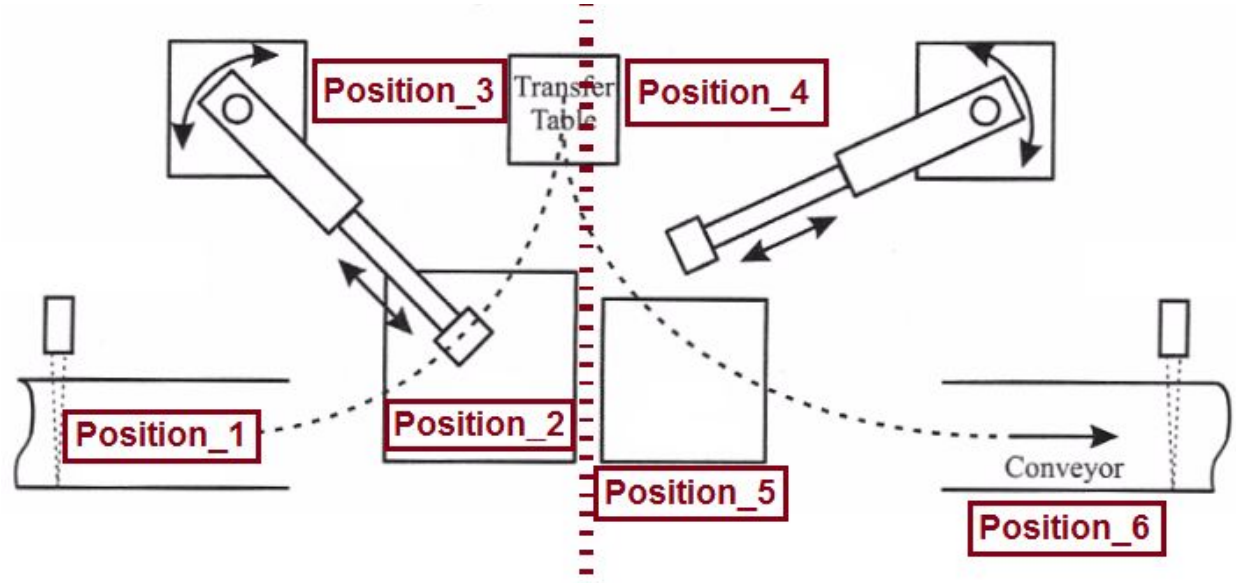


Figure 3: Specific positions for each robot arm. During the process, it is assumed that both robots are positioned at Position_3 and Position_4 at the start of the process for Robot_Arm_1 and Robot_Arm_2 respectively.

ROBOT ARM **ONE** Events:

- When Part_In sensor is activated, Conveyor_In motor stops
- As long as the Conveyor_In motor is stopped AND Part_In sensor memory is latched AND LS_1 is active, Vertical_Stepper_1 will move to Position_1 (90 degrees ClockWise)
- When Position_1 is reached (We will press buttons on arduino to advance stepper) it will activate TOF1_Sol_1 (which is a timer off delay "output") immediately extending Robot_Arm_1 (Sol_1)
- When TOF1_Sol_1 is done timing for one second, retract Robot_Arm_1 by cutting power to Sol_1 and powering Sol_2
- When LS_1 is activated and TOF_Sol_1 Done Timing bit is true, then rotate Vertical_Stepper_1 forty-five degrees CounterClockWise (for position 2)
- When Position_2 is reached and LS_1 is active, cut power to Sol_2
- When LS_1 is active and Sol_2 has power cut, then TOF2_Sol_1 timer off delay turns on immediately powering Sol_1
- When TOF2_Sol_1 is done timing for one second, retract Robot_Arm_1 by cutting power to Sol_1
- When TOF2_Sol_1 done bit and LS_1 are true, turn on TOF1_CNC_Machine off delay timer
- When TOF1_CNC_Machine ENABLE bit is true, run CNC_Fan_1 for One Minute
- When TOF1_CNC_Machine DONE bit is true, extend SOL_1 (robot arm to get part on lathe)

- When LS_2 is reached and TOF1_CNC_Machine DONE bit is true, turn on TOF1_Sol_2 timer off delay which is ONE SECOND
- When TOF1_Sol_2 timer off delay ENABLE BIT is true cut power to SOL_1 and power SOL_2 retracting part from lathe
- When LS_1 and NOT SOL_1, then turn forty-five degrees CounterClockWise (for position 2) on Vetical_Stepper_1 (look above)
- When position 3 is met, extend Cylinder_1 for 1 second, then retract.
- IF another part passes Part_In, the system should restart

ROBOT ARM **TWO** Events:

- When part is present (at transfer table) and LS_3 on cylinder 2, then the cylinder will extend (power sol_3)
- 1 second Time delay…
- When LS_4 is reached, retract cylinder 2 which is cutting power to SOL_3 and then powering SOL_4
- When LS_3 is reached and NOT Part_Present, rotate Vertical_Stepper_2 forty-five degrees clockwise
- When position_5 is reached, power SOL_3 for about a second (so add in TOF)
- When LS_3 and position 5 is reached, then power up CNC_Machine_2 for 1 minute 15 seconds using a Timer off delay
- When LS_3 and position 5 is reached, power up Conveyor_Out possibly using memory
- When timer off delay for CNC machine 2 is done, move Vertical_Stepper_2 fortyfive degrees counterclockwise to position 6
- When position 6 is reached, extend cylinder_2 by powering SOL_3
- time delay for 1 second, then after the second, retract the cylinder by taking away power from SOL_3 and powering SOL_4
- If time delay is reached, and position _6 is reached, return stepper to position_4
- If time delay is reached, and position _6 is reached, cut power to Conveyor_Out
- The system should repeat the process when there is another part at Part_Present

Section 3: List of Input and Output Devices

Physical Real World Inputs/Outputs

Card One		Card Two	
I:0/0	MASTER_START	O:1/0	CONVEYOR_IN
I:0/1	MASTER_STOP	O:1/1	CONVEYOR_OUT
I:0/2	ARDUINO_RESET	O:1/3	CNC_FAN_1
I:0/3	PART_IN	O:1/2	CNC_FAN_2
I:0/4	PART_OUT	O:1/4	ROBOT_ONE_RELAY_1
I:0/5	PART_PRESENT	O:1/5	ROBOT_ONE_RELAY_2
I:0/6	RESET_COUNT	O:1/6	ROBOT_TWO_RELAY_3
		O:1/7	ROBOT_TWO_RELAY_4
		O:1/8	ROBOT_ONE_RESET_RELAY
		O:1/9	ROBOT_TWO_RESET_RELAY

Virtual Outputs

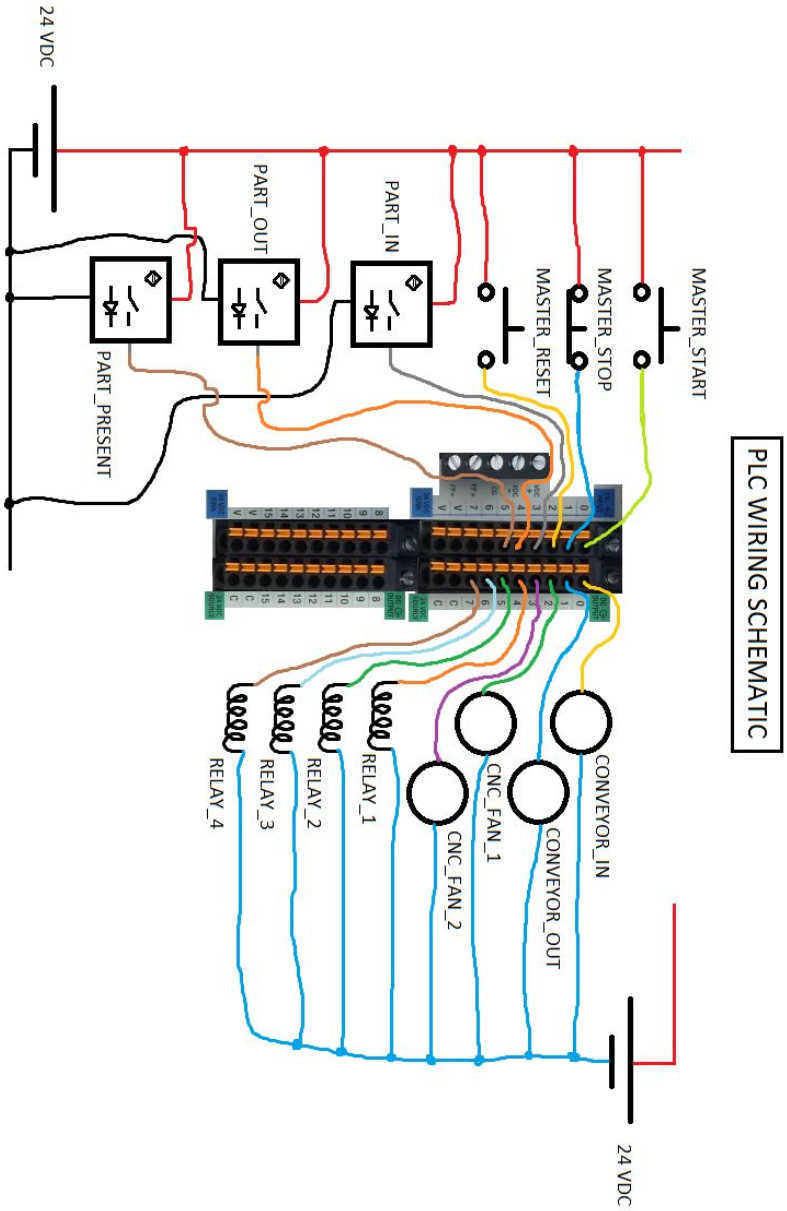
B3:0/0	POWER_RELAY
B3:0/1	TOTAL_CELL_PARTS_25
T4:0	CNC_TIME_1
T4:1	CNC_TIME_2
C5:0	TOTAL_PART_COUNTER
C5:0/RESET	TOTAL_PART_COUNTER/RESET

Section 4: IF/THEN Logic Statements for the PLC Program

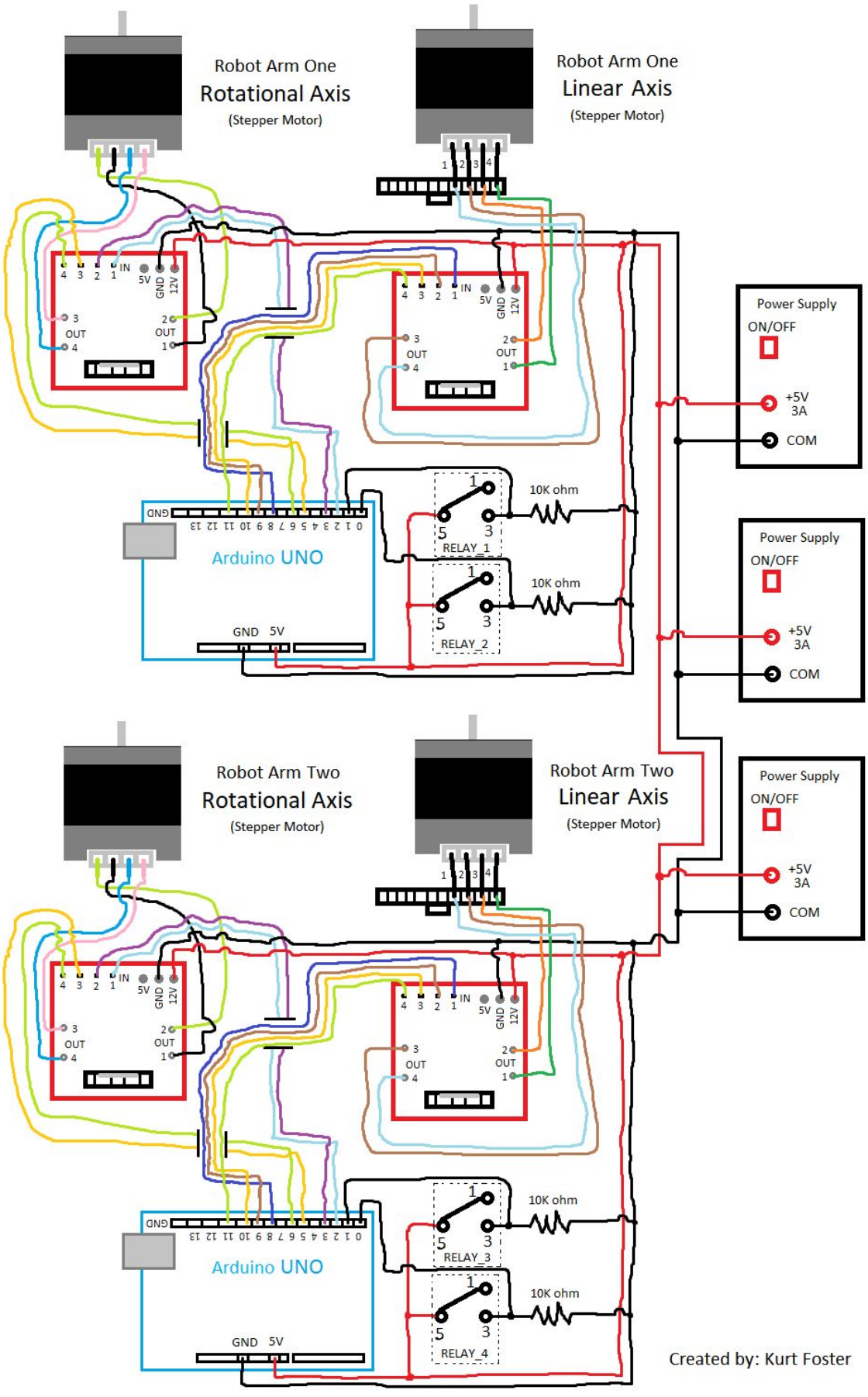
IF MASTER_STOP AND (MASTER_START OR POWER_RELAY) THEN POWER_RELAY
IF POWER_RELAY AND NOT PART_IN THEN CONVEYOR_IN
IF POWER_RELAY THEN CONVEYOR_OUT
IF POWER_RELAY AND (PART_IN OR CNC_TIME_1_TT_BIT) AND NOT (TOTAL_CELL_PARTS_25) THEN CNC_TIME_1
IF POWER_RELAY AND PART_IN THEN TOTAL_PART_COUNTER (CTU)
IF POWER_RELAY AND PART_OUT THEN TOTAL_PART_COUNTER (CTD)
IF PART_COUNTER.ACC EQUALS 25 THEN TOTAL_CELL_PARTS_25 internal relay
IF RESET_COUNT THEN RESET TOTAL_PART_COUNTER
IF POWER_RELAY AND (LOW_LIMIT of 5 > CNC_TIME_1_ACC_BIT of 0 > HIGH_LIMIT of 1000) THEN ROBOT_ONE_RELAY_1
IF POWER_RELAY AND (LOW_LIMIT of 7000 > CNC_TIME_1_ACC_BIT of 0 > HIGH_LIMIT of 15000) THEN CNC_FAN_1
IF POWER_RELAY AND (LOW_LIMIT of 16000 > CNC_TIME_1_ACC_BIT of 0 > HIGH_LIMIT of 17000) THEN ROBOT_ONE_RELAY_2
IF POWER_RELAY AND (PART_PRESENT OR CNC_TIME_2.TT) THEN CNC_TIME_2
IF POWER_RELAY AND (LOW_LIMIT of 5 > CNC_TIME_2_ACC_BIT of 0 > HIGH_LIMIT of 1000) THEN ROBOT_TWO_RELAY_3
IF POWER_RELAY AND (LOW_LIMIT of 7000 > CNC_TIME_2_ACC_BIT of 0 > HIGH_LIMIT of 15000) THEN CNC_FAN_2
IF POWER_RELAY AND (LOW_LIMIT of 16000 > CNC_TIME_2_ACC_BIT of 0 > HIGH_LIMIT of 17000) THEN ROBOT_TWO_RELAY_4
IF [POWER_RELAY AND (LOW_LIMIT 26000 > CNC_TIME_1_ACC > HIGH_LIMIT 27000 OR LOW_LIMIT 26000 > CNC_TIME_2_ACC > HIGH_LIMIT 27000)] OR (ARDUINO_RESET) THEN ROBOT_ONE&TWO_RESET_RELAY

Section 5: Wiring Diagram of the PLC and Arduino

PLC Wiring Diagram:



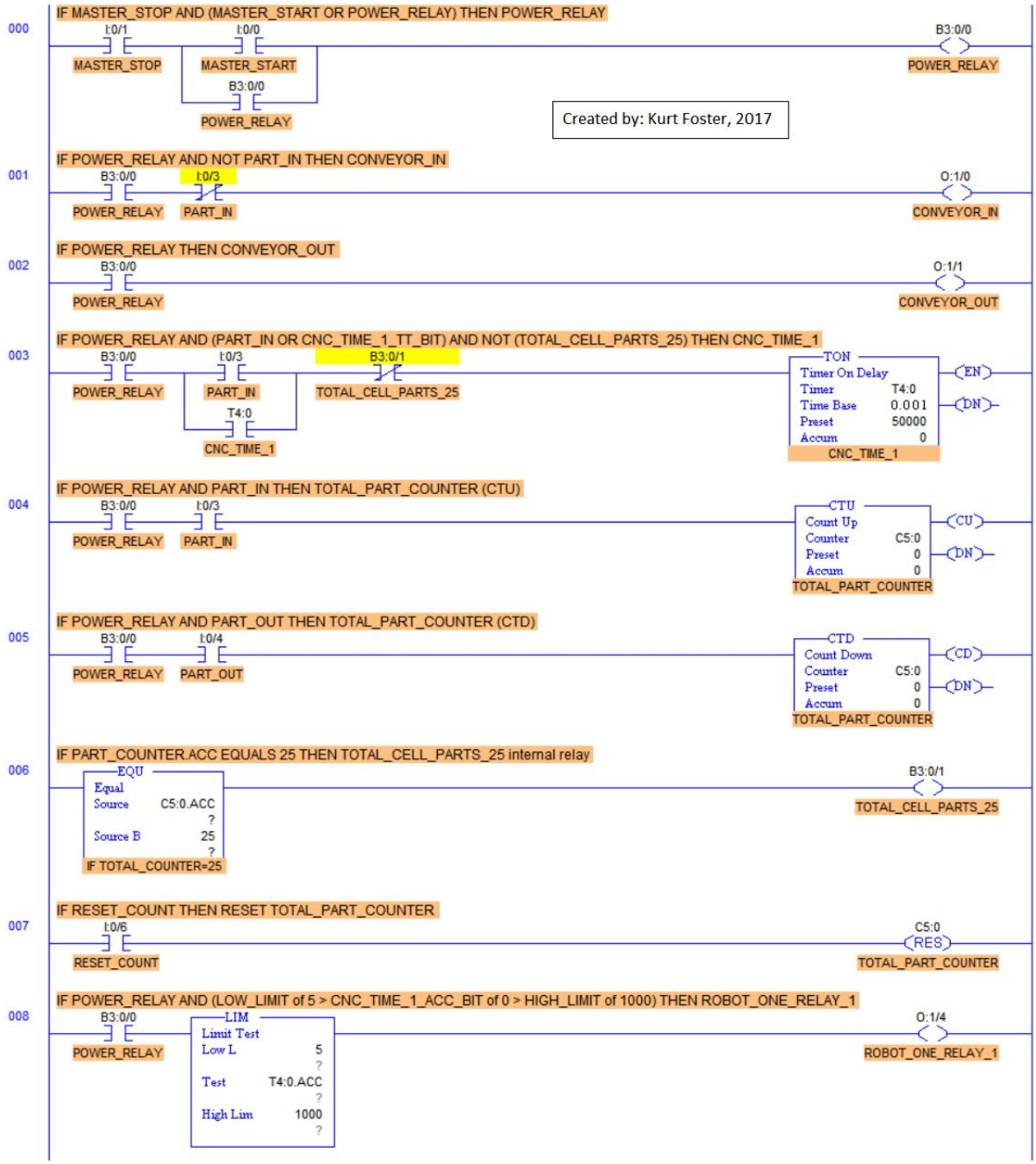
Arduino wiring diagram on the next page…



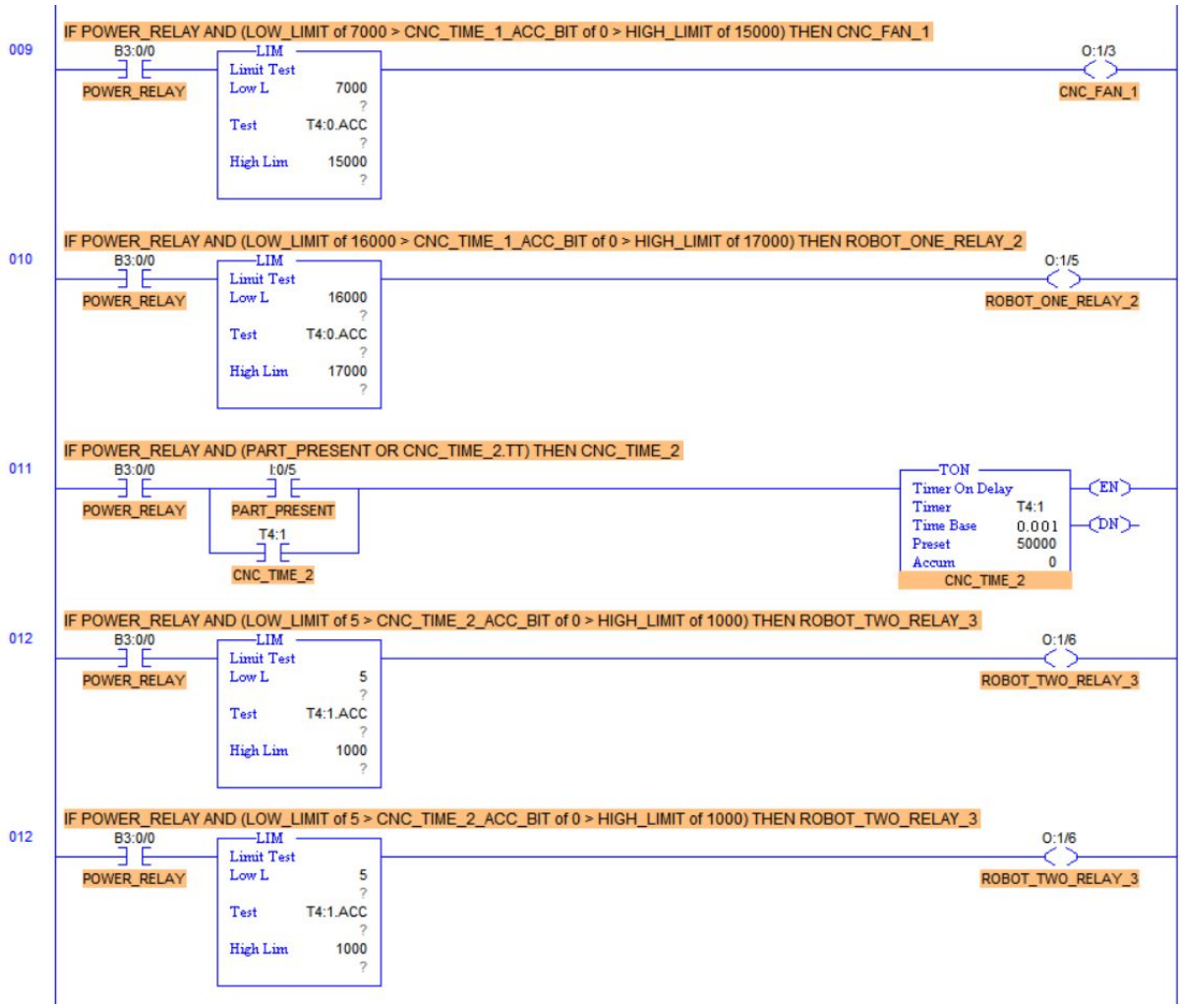
Created by: Kurt Foster

Section 6: PLC Ladder Logic and Arduino Code

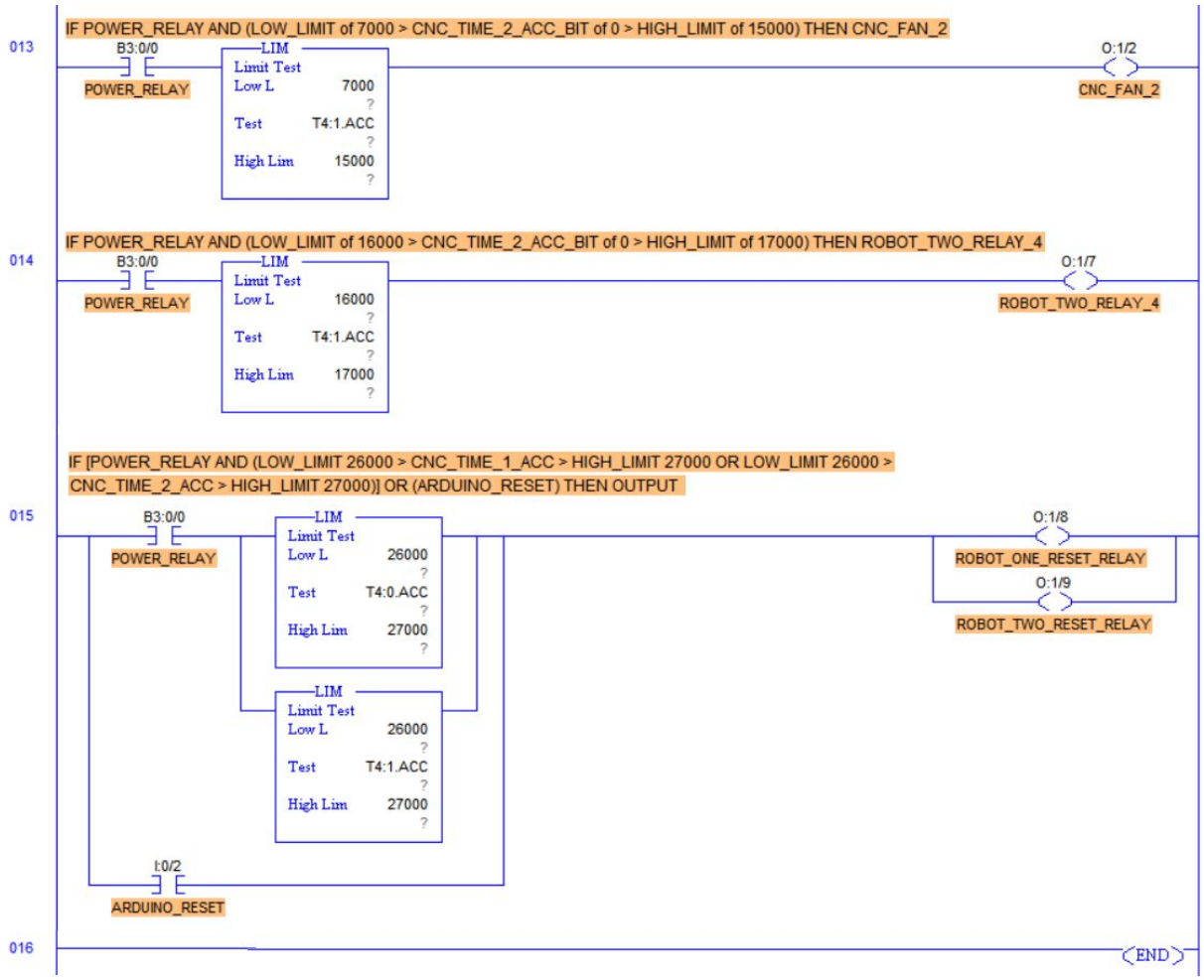
The following PLC program was created using the LogixPro Simulator. However, the program is designed to run on Allen-Bradley CompactLogix 5000 PLC's.



PLC code continued...



PLC code continued...



Next, arduino code that powered the 5V steppers will be shown...

BETA VERSION CODE FOR ROBOT ARM ONE

Code Description: The following code would be used for *single pushbutton manipulation of a single robot arm*. Every time you press the pushbutton momentarily, the robot would do the next sequence of events, then wait until the pushbutton is pressed again. The program implements the "while" function within each process.

```
#include <Stepper.h>
#define STEPS 200
Stepper stepper1(STEPS, 8, 9, 10, 11);  //  Extend/Retract Action Stepper
Stepper stepper2(STEPS, 0, 1, 2, 3);     //  Rotational Action Stepper
int pinButton = 6;
int LED = 13;

void setup()
{
pinMode(6, INPUT);            // sets the digital pin 6 as Input
pinMode(13, OUTPUT);       // sets the digital pin 13 as output
}

void loop()
{

int stateButton = digitalRead(6);          //read the state of the button

while (digitalRead(6) == LOW) {}       //wait until pushbutton
  if(stateButton == 1)           //if pushbutton is pressed

 {                                  //  NOTE: This bracket is required here for some reason...

            stepper2.setSpeed(50);          // Rotate Clockwise 90 deg
            stepper2.step(320);



while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(50);          //  Extend Gripper
            stepper1.step(-300);


while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(50);          // Retract Gripper to starting position
            stepper1.step(300);
```

```
while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper2.setSpeed(50);          // Rotate 45 degrees CounterClockWise
            stepper2.step(-160);


while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(50);          //  Extend Gripper
            stepper1.step(-300);


while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(50);          // Retract Gripper to starting position
            stepper1.step(300);


while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper2.setSpeed(50);          // Rotate 45 degrees CounterClockWise
            stepper2.step(-160);


while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(50);          //  Extend Gripper
            stepper1.step(-300);


while (digitalRead(6) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(50);          // Retract Gripper to starting position
            stepper1.step(300);
  }
}
```

BETA VERSION CODE FOR ROBOT ARM TWO

Code Description: The following code is similar to the code for robot arm one as shown above, however it makes use of the "stepSpeed" function. So by changing one number, you can change the speed of the stepper. The wafer handling machines seemed to have trouble operating above or below the range of 40 to 50, this issue was not resolved.

```
#include <Stepper.h>
#define STEPS 200
Stepper stepper1(STEPS, 8, 9, 10, 11);  //  Extend/Retract Action Stepper
Stepper stepper2(STEPS, 2, 3, 5, 6);      //  Rotational Action Stepper

int LED = 13;

int stepSpeed = 40;              //set all steppers full speed (DO NOT GO ABOVE 50, OR BELOW 40)
int inputPin = 7;

void setup()
{
pinMode(7, INPUT);            // sets the digital pin 6 as Input
pinMode(13, OUTPUT);        // sets the digital pin 13 as output
}

void loop()
{

int stateButton = digitalRead(inputPin);              //read the state of the button

while (digitalRead(inputPin) == LOW) {}           //wait until pushbutton
  if(stateButton == 1)           //if pushbutton is pressed

 {                                 //  NOTE: This bracket is required here for some reason...
  /*
---Sequence goes as follows---

            Extend gripper
            Retract gripper
            rotate arm 45 degrees Counterclockwise
            Extend gripper
            Retract gripper
            rotate arm 45 degrees counterclockwise
            Extend gripper
            Retract gripper
            rotate arm 90 degrees clockwise
            */
stepper1.setSpeed(stepSpeed);              //  Extend Gripper
            stepper1.step(-300);
```

```
while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(stepSpeed);          // Retract Gripper to starting position
            stepper1.step(300);

while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper2.setSpeed(stepSpeed);          // Rotate 45 degrees CounterClockWise
            stepper2.step(-160);

while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(stepSpeed);          //  Extend Gripper
            stepper1.step(-300);

while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(stepSpeed);          // Retract Gripper to starting position
            stepper1.step(300);


while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper2.setSpeed(stepSpeed);          // Rotate 45 degrees CounterClockWise
            stepper2.step(-160);


while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(stepSpeed);          //  Extend Gripper
            stepper1.step(-300);

while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper1.setSpeed(stepSpeed);          // Retract Gripper to starting position
            stepper1.step(300);

while (digitalRead(inputPin) == LOW) {}
  if(stateButton == 1)
            stepper2.setSpeed(stepSpeed);          // Rotate 90 degrees ClockWise
            stepper2.step(320);

  }
}
```

FINAL CODE FOR ROBOT ARM ONE

```
#include <Stepper.h>
#define STEPS 200
Stepper stepper1(STEPS, 8, 9, 10, 11);  //  Extend/Retract Action Stepper
Stepper stepper2(STEPS, 2, 3, 5, 6);    //  Rotational Action Stepper

int LED = 13;

int stepSpeed = 40;         //set all steppers full speed (DO NOT GO ABOVE 50, and below 40)
int inputPin1 = 7;
int inputPin2 = 4;
int OutputPin1 = 12;
int OutputPin2 = 13;

void setup()
{
pinMode(inputPin1, INPUT);          // sets the digital pin 7 as Input
pinMode(inputPin2, INPUT);          // sets the digital pin 4 as Input
pinMode(OutputPin1, OUTPUT);
pinMode(OutputPin2, OUTPUT);
pinMode(13, OUTPUT);          // sets the digital pin 13 as output

}


void loop()
{
  /*

---Sequence goes as follows:

   Extend gripper
   Retract gripper
   rotate arm 45 degrees Counterclockwise
   Extend gripper
   Retract gripper
   rotate arm 45 degrees counterclockwise
   Extend gripper
   Retract gripper
   rotate arm 90 degrees clockwise

   */


if((digitalRead(inputPin1) == HIGH))
 {

   stepper2.setSpeed(stepSpeed);          // Rotate 90 degrees ClockWise
```

```
    stepper2.step(320);

    stepper1.setSpeed(stepSpeed);        //  Extend Gripper
    stepper1.step(-300);

    stepper1.setSpeed(stepSpeed);         // Retract Gripper to starting position
    stepper1.step(300);


    stepper2.setSpeed(stepSpeed);         // Rotate 45 degrees CounterClockWise
    stepper2.step(-160);

    stepper1.setSpeed(stepSpeed);        //  Extend Gripper
    stepper1.step(-300);

  }
if((digitalRead(inputPin2) == HIGH))
  {

    stepper1.setSpeed(stepSpeed);         // Retract Gripper to starting position
    stepper1.step(300);

    stepper2.setSpeed(stepSpeed);         // Rotate 45 degrees CounterClockWise
    stepper2.step(-160);

    stepper1.setSpeed(stepSpeed);        //  Extend Gripper
    stepper1.step(-300);

    stepper1.setSpeed(stepSpeed);         // Retract Gripper to starting position
    stepper1.step(300);

    digitalWrite(OutputPin1, LOW);

  }

}
```

FINAL CODE FOR ROBOT ARM TWO

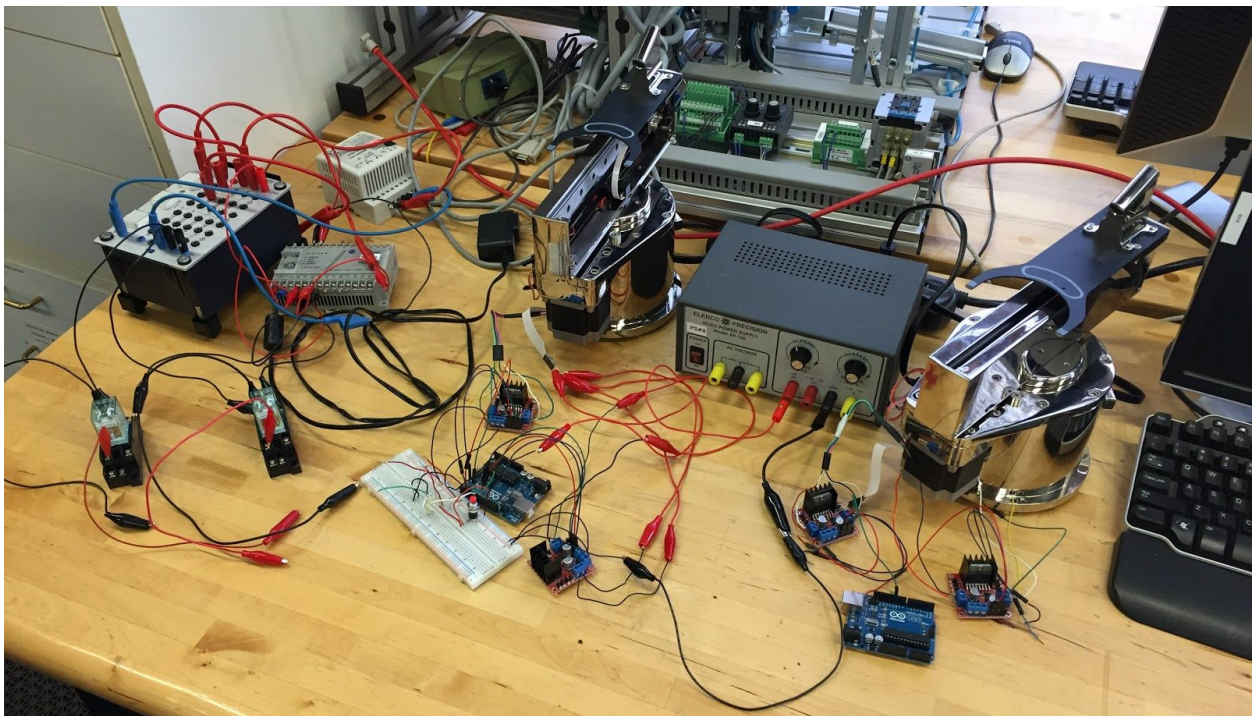
```
#include <Stepper.h>
#define STEPS 200
Stepper stepper1(STEPS, 8, 9, 10, 11);  //  Extend/Retract Action Stepper
Stepper stepper2(STEPS, 2, 3, 5, 6);    //  Rotational Action Stepper

int LED = 13;

int stepSpeed = 40;         //set all steppers full speed (DO NOT GO ABOVE 50, and below 40)
int inputPin1 = 7;
int inputPin2 = 4;
int OutputPin1 = 12;
int OutputPin2 = 13;

void setup()
{
pinMode(inputPin1, INPUT);          // sets the digital pin 7 as Input
pinMode(inputPin2, INPUT);          // sets the digital pin 4 as Input
pinMode(OutputPin1, OUTPUT);
pinMode(OutputPin2, OUTPUT);
pinMode(13, OUTPUT);          // sets the digital pin 13 as output

}

void loop()
{
  /*

---Sequence goes as follows:

   Extend gripper
   Retract Gripper to starting position
   Rotate 45 degrees CounterClockWise
   Extend gripper
   Retract gripper
   rotate arm 45 degrees counterclockwise
   Extend gripper
   Retract gripper
   rotate arm 90 degrees clockwise

   */

if((digitalRead(inputPin1) == HIGH))
  {

   stepper1.setSpeed(stepSpeed);        //  Extend Gripper
   stepper1.step(-300);
```

```
    stepper1.setSpeed(stepSpeed);          // Retract Gripper to starting position
    stepper1.step(300);

    stepper2.setSpeed(stepSpeed);          // Rotate 45 degrees CounterClockWise
    stepper2.step(-160);

    stepper1.setSpeed(stepSpeed);        //  Extend Gripper
    stepper1.step(-300);

  }
if((digitalRead(inputPin2) == HIGH))
  {

    stepper1.setSpeed(stepSpeed);          // Retract Gripper to starting position
    stepper1.step(300);

    stepper2.setSpeed(stepSpeed);          // Rotate 45 degrees CounterClockWise
    stepper2.step(-160);

    stepper1.setSpeed(stepSpeed);        //  Extend Gripper
    stepper1.step(-300);

    stepper1.setSpeed(stepSpeed);          // Retract Gripper to starting position
    stepper1.step(300);

    stepper2.setSpeed(stepSpeed);          // Rotate 90 degrees ClockWise
    stepper2.step(320);

    digitalWrite(OutputPin1, LOW);

  }
}
```

Section 7: Photos of the System Wired Together

Here is the prototype wired system done early on. This was to see if a PLC could tell an arduino what to do and vice-versa. You can also see the wafer handling machines that were used for operation. They were just donated to the college, and I was the first to get them up and running. The little red boards next to the arduino's are motor drivers rated to 3A. There is one for each stepper motor, who's running amperage was 2A.

Demonstration YouTube Link: https://youtu.be/VDCZo-pmgzg



As you can probably tell, I was already running out of room. This project didn't require organizing the wire system, but if I had extra time, I would do that next. Making the system more organized makes it easier to troubleshoot problems.

Here is the system all “organized” with the two 120V CNC machine “fans” in the front. Start is the green, stop is the red. The blue button is the reset.

This next picture was a little more into the process. 120V fans were replaced with 24V fans, and I added extra relays to give reset functionality to the arduinos via the PLC.