Configuration of a PLC Controlled Articulated Robot for Autonomous Vision Inspection Applications

Maged Mikhail, PhD[1], and Winston Sealy, PhD[2]
[1]Purdue University Northwest, Hammond, USA, mmikhail@pnw.edu
[2]Purdue University, West Lafayette, USA, dwsealy@purdue.edu

***Abstract*– As manufacturing and industry continues its growth to meet demands, new and more efficient industrial systems must be developed and validated. As such, machine vision systems, for example, can be integrated with robotic systems for improved results. These robots can autonomously identify trained components in conjunction with a Programmable Logic Controller (PLC) to perform industrial tasks that are more efficient. This research project resulted in the complete integration and configuration of a vision system and articulated robot, that was controller by an external PLC for autonomous product detection.**

***Keywords*-- Articulated Robot, Ladder Logic, Vision System, PLC.**

I. Introduction

Developing countries, such as Latin America and the Caribbean countries, must be creative and innovative in addressing manufacturing efficiencies [1]. Automation has been a proven solution to many cost mitigations within manufacturing facilities. Many have realized significant cost reductions through automation with robots and PLCs serving as the main technology. More so, the integration of the robot and PLC can further enhance production. The PLC is a widely used computer device for reducing production costs, while improving reliability and flexibility of manufacturing processes [2]. Furthermore, the PLC can be programmed with limited knowledge of computer programming [3]. Manufacturer are always in search of methods and best practices that can be adopted to improve overall performance and reduce the costs associated in making products. Considering that most production processes utilize some form of automation, this project, explores the concept of a PLC controlled articulated robot arm for use in autonomous inspection of products.

II. System Integration

The system integration and configuration consisted of the following [4]:

- Configuring and wiring the devices
- Mapping the Input/output (I/O) on the controller to the connections
- Sending the signal from PLC using ladder logic program

In this project, we integrated and configured the Fanuc LR Mate 200iD articulated robot, iRVision camera, and Allen-Bradley MicroLogix 1100 PLC. The LR Mate 200iD is a compact six-axis mini articulated robot with the approximate size and reach of a human arm that is compatible with the iRVision camera system. The Allen-Bradley MicroLogix 1100 controller, combines input and output modules for automating the system.

Configuring and wiring the devices can be achieved by two methods. The PLC can communicate to the robot via ethernet connection or by a digital (I/O) connector bus. This project utilized digital I/O connections, because the selected robot was not equipped the required ethernet adapter card needed for communication.

The digital inputs of the PLC were connected to the digital outputs on the robot, and the digital inputs of the robot connected to the digital outputs of the PLC. Fig. 1 illustrates the I/O connector bus with Allen-Bradley PLC.

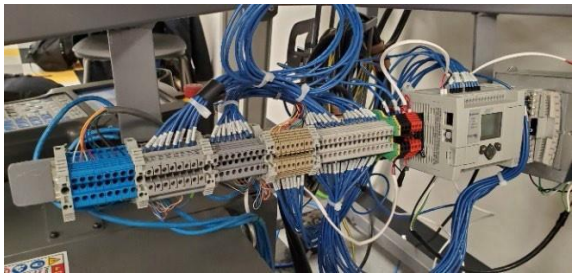

Fig. 1 PLC Wiring Bus

III. PLC to Robot Communication

Following the physical connections, we ensure that the PLC communicated with the robot. The verification was done through observing the Digital Input (DI) page on the Teach Pendant, while toggling output bits on the PLC. The process was reversed to ensure that changing Digital Output (DO) values on the teach pendant affect values on the PLC input data file.

Once the communication between the PLC, and the Robot were verified, the next step was to prepare the robot for

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2022.1.1.347
ISBN: 978-628-95207-0-5 **ISSN:** 2414-6390

automatic control via the PLC. The following settings were enabled for remote automated control: Enable UI signals, START for CONTINUE only, CSTOPI for ABORT, abort all programs by CSTOPI, and PROD_START required PNSTROBE (fig 2).

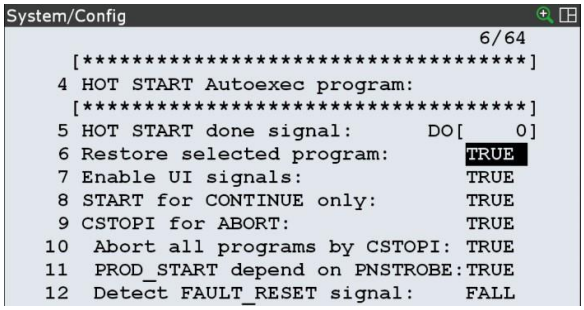

Fig. 2 System / Configuration Menu

IV. Teach Pendant Configuration

In order to successfully communicate via the I/O bus, the teach pendant must be configured to accept incoming signals and relate them to specific commands such as start, stop, hold, etc. Also, these settings must be reverse in order to run any program from the teach pendant. It must be noted that remote control to run a program while the teach pendant is configured for local programming is won't work, and likewise, cannot locally run a program while the teach pendant is configured for remote control.

In order to fully configure the teach pendant I/O, both digital input and outputs were configured, as well as user inputs and outputs. To configure the digital inputs, we navigated the menu by pressing the following keys (table 1):

1. [MENU] → [I/O]
2. Press the [F1] hard-key or tap [TYPE] on the touchscreen
3. Navigate to [DIGITAL] and press the [Enter] hard key
4. Press the [F2] hard-key or tap CONFIG on the touch screen
5. The display should indicate "Digital In" near the top of the screen. If not, press the IN/OUT button on the touchscreen or the [F3] hard key (Fig. 5)

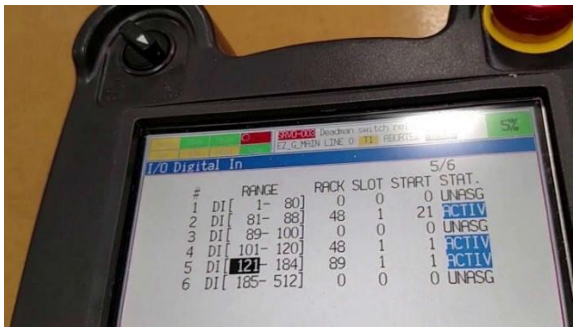

Fig. 3 Digital Input Configuration – Teach Pendant

These configuration settings will cause the inputs to be related to the digital inputs on the controller, allowing them to be accessed and read from within the controller program. The last column, "Status", may not show as "Active" until the controller is power cycled. This is normal and can be done after all configurations are done to avoid turning the controller on and off unnecessarily. Please note, these changes will not become active until the controller is power cycled.

Configuring the user inputs was a similar procedure to digital inputs. We navigated to the user inputs screen:

1. [MENU] → [I/O]
2. Press the [F1] hard-key or tap [TYPE] on the touchscreen
3. Navigate to [UOP] and press the [Enter] hard-key
4. Press the [F2] hard-key or tap CONFIG on the touch screen
5. The display should indicate "UOP In" near the top of the screen. If not, press the IN/OUT button on the touchscreen or the [F3] hard-key

TABLE 1: Digital Input

#	Range	Rack	Slot	Start
1	UI [1 – 1]	48	1	1
2	UI [2 – 2]	48	1	2
3	UI [3 – 3]	48	1	3
4	UI [4 – 4]	48	1	4
5	UI [5 – 5]	48	1	5
6	UI [6 – 6]	48	1	6
7	UI [7 – 7]	48	1	7
8	UI [8 – 8]	48	1	8
9	UI [9 – 12]	48	1	9
10	N/A	N/A	N/A	N/A
11	UI [17 – 17]	48	1	14
12	UI [18 – 18]	48	1	15

**Note: N/A indicates that configuration is not necessary*

Properly configuring the user inputs is particularly important, because it gives access to features such as remote state, hold, fault resets, etc. These features can be viewed after configuration by pressing the CONFIG button, or by pressing [F2] hard-key.

After setting up digital and user inputs, the next step was the outputs. Again, configuration follows the same procedures as the previous configurations. We navigated the digital input configuration screen on the teach pendant (table 2):

1. [MENU] → [I/O]
2. Press the [F1] hard-key or tap [TYPE] on the touchscreen
3. Navigate to [DIGITAL] and press the [Enter] hard-key
4. Press the [F2] hard-key or tap CONFIG on the touch screen
5. The display should indicate "Digital Out" near the top of the screen. If not, press the IN/OUT button on the touchscreen or the [F3] hard-key

TABLE 2: User Input

#	Range	Rack	Slot	Start
1	N/A	N/A	N/A	N/A
2	DO [81 - 84]	48	1	21
3	N/A	N/A	N/A	N/A
4	DO [101 - 120]	48	1	1
5	N/A	N/A	N/A	N/A

**Note: N/A indicates that configuration is not necessary*

After configuring the digital outputs, they were tested by pressing the MONITOR button or the [F2] hard-key. Navigate to digital output 101. Digital outputs 101 through 120 will trigger and be able to be detected by the PLC. Fig. 4 shows the digital outputs that can be manually triggered for testing. Of course, these outputs can also be triggered from within a robot program as well.

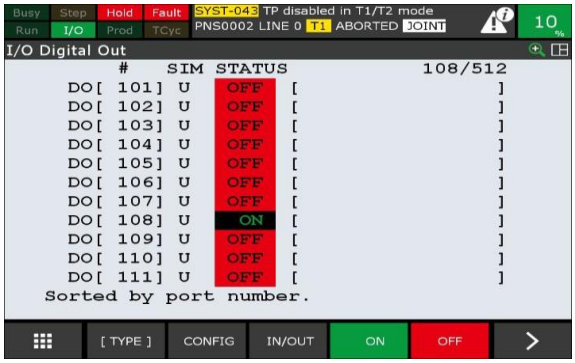

Fig. 4 Digital Outputs

The final configuration that must be done is through the user outputs configuration screen. Configuring these outputs allows the digital outputs to be used from within a robot program to trigger PLC inputs. To navigate to the configuration screen:

1. [MENU] → [I/O]
2. Press the [F1] hard-key or tap [TYPE] on the touchscreen
3. Navigate to [UOP] and press the [Enter] hard-key
4. Press the [F2] hard-key or tap CONFIG on the touch screen
5. The display should indicate "UOP Out" near the top of the screen. If not, press the IN/OUT button on the touchscreen or the [F3] hard key

The following changes must be made to correctly configure the user outputs (table 3):

Table 3: User Outputs

#	Range	Rack	Slot	Start
1	UO [1 – 1]	48	1	21
2	N/A	N/A	N/A	N/A
3	UO [6 – 6]	48	1	22
4	N/A	N/A	N/A	N/A
5	UO [9 – 9]	48	1	23
6	UO [10 – 10]	48	1	24
7	N/A	N/A	N/A	N/A

**Note: N/A indicates that configuration is not necessary*

After configuring the user outputs, the teach pendant was set up to fully accept inputs as well as produce outputs to the PLC. This established the communication between the controller and the PLC; however, at this point the PLC was unable to control critical aspects of the robot through I/O. Controls such as Hold, Start, and fault reset needed to be configured to be recognized through inputs.

V. Robot Vision Setup

To utilize the vision system, a set of 3D printed shapes were made to be detected by the camera. They were printed with white filament to better contrast the black background used for the base. This allowed for improved object detection. The shapes are shown in Figures 5 and 6 to represent good and bad products, respectively.

Fig. 5 Good (Taught) Part

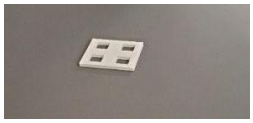

Fig. 6 Bad Part

To properly detect an object, with reliability and repeatability, the vision system was calibrated to the appropriate ratio of pixels to millimeters. Fig. 7 illustrates the proper placements of the calibrations grid in respect to the camera and the robot. It was important to ensure that the camera was focused to the grid in order to accurately detect the parts for training.

Ensure the iRVision software was updated and configured. In the Vision Setup menu, we created a new camera setup. We used a 30mm grid spacing as pictured in fig. 7. The calibration process located all the tracking points to determine the physical distance for the camera.

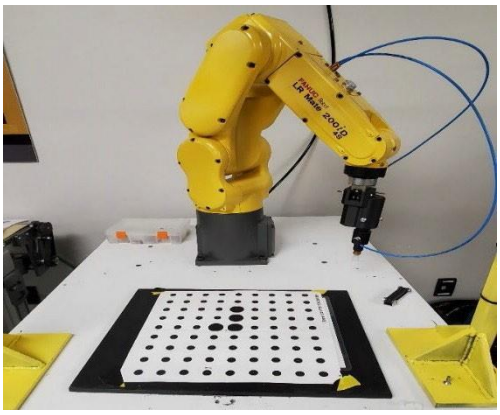

Fig. 7 Calibration Grid Placement

After calibration, the system was ready to be trained to detect objects. For the project, a simple pass/fail model was used: if the part is correct, a Boolean True was set, otherwise a Boolean False was recorded. The system was taught to detect based on a GPM locator tool. The locator tool discriminated the appropriate edges of the part. An output was produced based on the part's score. The variable "V1" was assigned to the GPM score parameter and was true if it was greater than 70% confidence (fig. 8). Once trained, a single view inspection process was utilized to detect the part.

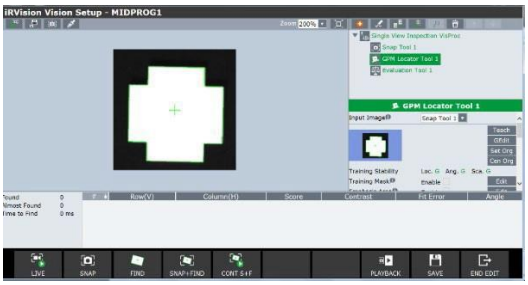

Fig. 1 Part Learning

VI. Programming

Programming consisted of the main PLC program and two robot specific programs. The main PLC program performed all the triggering and processing commands of the system, while the two robot specific programs provided instructions for inspection and part selection. The main program was written using ladder logic on the PLC, while the two sub-programs resided on the robot. Fig. 9 reveals the programming flow of the process, from start to calling of the subprograms.

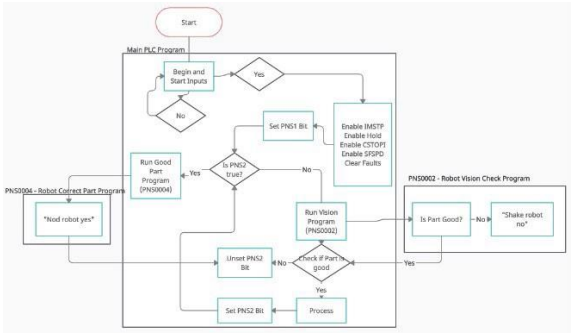

Fig. 9 Project Flow-Chart

The PLC was the main driver, and looped indefinitely while making decisions based on the inputs from the robot's vision system output. If the detected part was correct, the PLC shifted function to the correct part program (PNS0004). However, If the detected part was bad or there were no parts to detect, then the PLC would shift function back to the vision checking system to continue inspecting parts. The following sections explain each program in detail.

A. *Robot PNS Programs*

The robot programs followed a very strict naming convention. Since the programs are triggered remotely, by the PLC, a simplified naming convention needed to be implemented. To achieve this, a four-bit code was sent to the controller via four separate PLC output lines. These four bits corresponded to PNS1, PNS2, PNS3, and PNS4. Using a single or combination of bits, the correct sub-programs were executed. Enabling the PNS1 bit on the PLC, for example, executed the program PNS0001. Similarly, enabling both PNS1 and PNS2 bits executed the program PNS0003. The PNS bit selection was determined from binary math: PNS1 is 2^0, PNS2 is 2^1, and so on. Therefore, $2^0 + 2^1 = 3$, for PNS0003. This allows for a total of 15 programs to be selected with 4 bits, or even more by enabling more PNS bits in the configuration.

There is also the option to configure a "base" number for the PNS programs. This adds the base number onto the received PNS bits to select a program. If PNS1 was active and the base was 100, for example, then PNS0101 would be selected. In this project, the base was set to 1, where each program received a single unit offset.

B. *PNS0002 – Robot Vision Program*

The PNS0002 (fig. 10) program was responsible for handling the processing of the robot vision, as well as returning the results of the inspection to the main PLC program.

```
PNS0002                                          1/30
  1:  UFRAME_NUM=2
  2:  UTOOL_NUM=2
  3:
  4:
  5:  DO[108]=OFF
  6:  DO[107]=OFF
  7:  !7 - good 8 - bad
  8:
  9:  VISION RUN_FIND 'MIDPROG1'
 10:  VISION GET_PASSFAIL 'MIDPROG1'
      :  R[5]
 11:
 12:  IF (R[5]=1) THEN
 13:  DO[107]=ON
 14:  JMP LBL[5]
 15:  ELSE
 16:  DO[107]=OFF
 17:  ENDIF
 18:  IF (R[5]=2) THEN
 19:  DO[108]=ON
 20:  ELSE
 21:  DO[108]=OFF
 22:  ENDIF
 23:
 24:J  P[1] 20% CNT100
 25:J @P[2] 20% CNT100
 26:
 27:  LBL[5]
 28:  DO[106]=PULSE,0.5sec
 29:  R[5]=0
[End]
```

Fig. 10 PNS0002 Program

The line-by-line explanation of the program is as follows:

Line 1: Set the user frame to frame 2
Line 2: Set the tool frame to frame 2
Line 5: Reset the bad case output bit (unused)
Line 6: Reset the good case output bit
Line 9: Run the vision program defined in
Line 10: Get the result of the vision program and store it in register [5]
Line 12: If the vision program detected the part:
Line 13: Set the good output bit to true
Line 14: Jump to label 5, line 27
Line 15: If the vision program did not detect the part:

Line 16: Set the good output bit to false
Line 18-22 are not relevant to the current project and can be omitted.
Line 24: Move the head back and forth, mimicking a "no" gesture
Line 25: Movement command described in line 24
Line 27: This is the location the program jumps to if a part is detected, skipping lines 15-25
Line 28: Pulse output 106, resetting the program select in the PLC program
Line 29: Resets register 5 to 0

When the robot identified a good part, a bit signal was returned to the PLC, which, in turn, enabled a separate PNS program. The PNS strobe and production start input bits must also be toggled on the controller for proper communications. Since triggering is controlled by the PLC, the controlling rung must be reset, and this was accomplished by **Line** 28 of the PNS0002 program. This will be explained further in the PLC program explanation.

C. PNS0004 – Robot Good Part Program

The PNS0004 (fig.11) was triggered if PNS0002 detected a good part. As noted, the triggers come from the PLC and not from any internal controller programs.

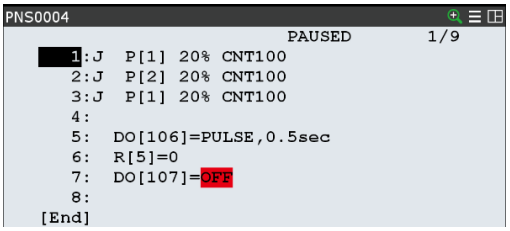

Fig. 11 PNS0004 Good Part Program

Lines 1-3: These lines produce a "yes" like gesture using the end effector
Line 5: Pulses output 106, resetting the PLC program trigger
Line 6: Resets register 5 to 0
Line 7: Sets the "good part" bit to 0, allowing for another vision check

As mentioned, it is necessary to reset the program initiation triggers on the PLC to properly trigger additional programs. Disabling the "good part" bit allowed for the PLC to run the vision program repeatedly during inspection.

D. Main PLC Program

The heart of the project lies within a ladder logic program written for the Micro Logix 1100 PLC controller. This ladder program was written using the software Logix 500, and contains all control logic that allowed the robot to detect, and subsequently act upon, the vision detection results. The program controlled the *start, hold, fault reset, program select*, and program triggering functions of the robot controller. The comments within the program were removed for better readability when describing the rungs. Each rung is described separately below.

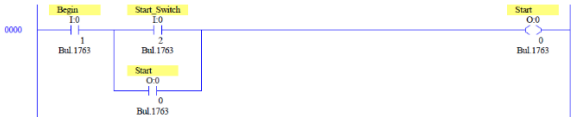

Fig. 12 Rung 0000

Rung 0 (fig. 12)**:** This is the first rung of the program. It waits for both a "Begin" input as well as a "Start switch" input. The start switch will latch itself upon activation, and the begin contact will break this latch if it is set to false. Both of these contacts were activated via forces when testing the program, but can easily be assigned to inputs on the PLC and wired to buttons or switches.

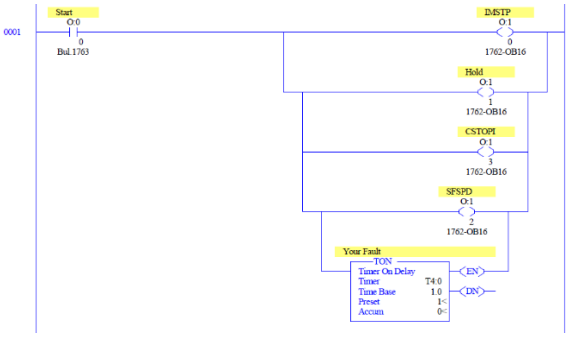

Fig. 13 Rung 0001

Rung 1 (fig. 13)**:** This rung is triggered after the start contact has been latched. It sets several outputs, which in turn enable the following parameters on the robot: IMSTP, Hold, CSTOPI, and SFSPD. All of these are required to run a robot program. A one second timer is also energized for clearing faults. The fault clearing must be toggled, but also enabled for a set amount of time. Therefore, a TON timer was used.

Fig. 14 Rung 0002

Rung 2 (fig. 14)**:** This rung sets fault reset bit to true while the TON timer in rung 1 is timing. This acts to toggle the fault reset and ensure that it does not remain on for the duration of the program.

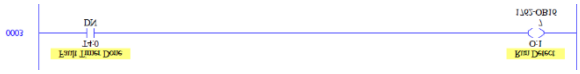

Fig.15 Rung 0003

Rung 3 (fig. 15)**:** After the fault resetting timer is done timing, the program continues. "Run Detect" enables the program to select PNS bits for program triggering.

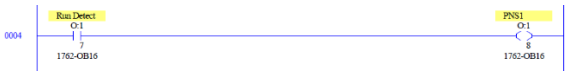

Fig. 16 Rung 0004

Rung 4 (fig. 16)**:** If the faults have been reset, meaning that "Run Detect" is active, then the PNS1 program selector bit is turned active. This bit remains active for the duration of the program.

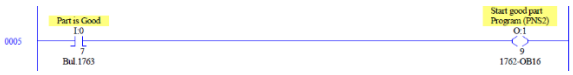

Fig.17 Rung 0005

Rung 5 (fig. 17)**:** This rung adds an addition PNS selector bit if the "Part is Good", input is true. This input is tied to the robot controller output 107, meaning that if a detected part is good, then this bit is true. If this is the case, then the current PNS value is PNS1 + PNS2. otherwise, it is PNS1.

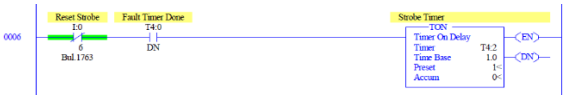

Fig. 18 Rung 0006

Rung 6 (fig. 18)**:** This rung controls the PNS strobe function of the controller. The PNS strobe essentially "sends" the PNS bits to the controller, for sub-program selection. Because it is a strobe, it must be toggled and therefore a timer is used. "Reset Strobe" is attached to robot controller output 106. This output is pulsed at the end of both robot programs. This allows the strobe timer to be reset, which in turn allows a new program or existing program to be selected and ran.

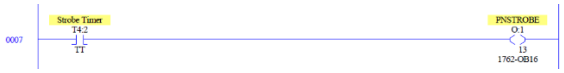

Fig. 19 Rung 0007

Rung 7 (fig. 19)**:** As mentioned in Rung 6, this rung enables the PNSTROBE bit while the strobe timer is timing. PNSTROBE points to User input # 12 on the robot controller.

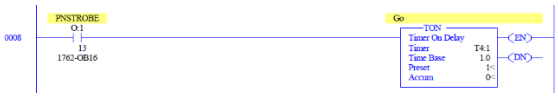

Fig. 20 Rung 0008

Rung 8 (fig. 20)**:** While the strobe timer is timing, the "Go" timer is also triggered. This timer controls the start and production start functions of the robot controller. Two separate timers were used to create a very small (milliseconds) delay between setting and sending the PNS bits and starting the program. This ensured that the data was registered.

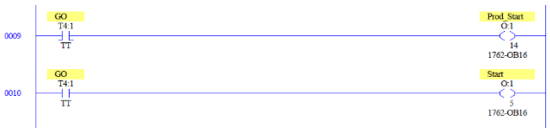

Fig. 21 Rung 0009 and Rung 0010

Rungs 9 and 10 (fig. 21): These rungs run while the Go timer is timing, to create a toggle for "*Prod Start*" and "*Start*". Although similarly named, these outputs are different from each other. Both of these start commands must be triggered to start the selected PNS program. *Prod Start* points to User Input #13 on the robot, while Start points to User Input #4.

All in all, the program will first trigger the vision system and request an input based on the output of that process. If a part is detected, it will set an additional PNS bit the next time the ladder program is ran. While this PNS bit is active, the vision program is bypassed in favor of the "Yes" confirmation program, which in turn resets the detected part bit. This reset effectively resets the additional PNS bit, allowing for the vision program to be ran again. This process loops indefinitely until power is lost, or the "*Begin*" contact is de-energized.

VII. Conclusion

Communication between an articulated robot and an external PLC is simple in concept but challenging in practice. Several configurations must be made to allow for communication as well as to allow for remote control of critical functions. Nevertheless, the benefits outweigh the requirements of this application. Allowing a PLC to control a robot provides a much greater range of control. Using a PLC to control a robot allows for the use of an HMI, external indicators and controls, as well as internet connectivity without exposing the robot to a potentially dangerous network environment.

In conclusion, this project was a great exercise in problem solving and systems integration. It provided great opportunities for students to grow and learn new and unfamiliar systems. Students can gain valuable insights on integration and configuration of an engineering system. The project provided a deeper knowledge of programmable logic controllers, industrial robotics, and vision systems. All in all, the project was a key stepping stone to further autonomous solution building, and in thinking of problems systematically. In short, the project will expose students to a systematic way of thinking that can be valuable for solving complex engineering problems.

REFERENCES

[1] Sealy, W. (2012). A Finite Element Study of a Patient Lift Conceptual Design. *In Proceedings of the 10th Latin American and Caribbean Conference for Engineering and Technology– Megaprojects: Building Infrastructure by fostering engineering collaboration, efficient and effective integration and innovative planning, Panama City, Panama, Session R-2C Engineering Design III.*

[2] Dorjee, R. G. (2014). PLC and Fuzzy Logic Control of a Variable Frequency Drive. International Journal of Engineering Trends and Technology (IJETT), 16(4), 2231-5381.

[3] Maha M. Lashin, (2014). Different Applications of Programmable Logic Controller (PLC). International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), 4(1), 27-32.

[4] A. Sergeyev, N. Alaraje, S. Parmar, S. Kuhl, V. Druschke, and J. Hooker, (2017). Promoting industrial robotics education by curriculum, robotic simulation software, and advanced robotic workcell development and implementation, *Annual IEEE International Systems Conference (SysCon)*.

[5] Ka'bi A. (2021). Energy Consumption Management Using Programmable Logic Controllers (PLC's), *IEEE Technology & Engineering Management Conference - Europe (TEMSCON-EUR), 2021, pp. 1-6, doi: 10.1109/TEMSCON-EUR52034.2021.9488607*.