THE UNIVERSITY OF MICHIGAN-DEARBORN

SCHOOL OF ENGINEERING

INDUSTRIAL AND MANUFACTURING SYSTEMS ENGINEERING DEPARTMENT

IMSE/CIS 381: INDUSTRIAL ROBOTICS

ASSIGNMENT #4

# Robot Programming

**Student Name: Demetrius Johnson**

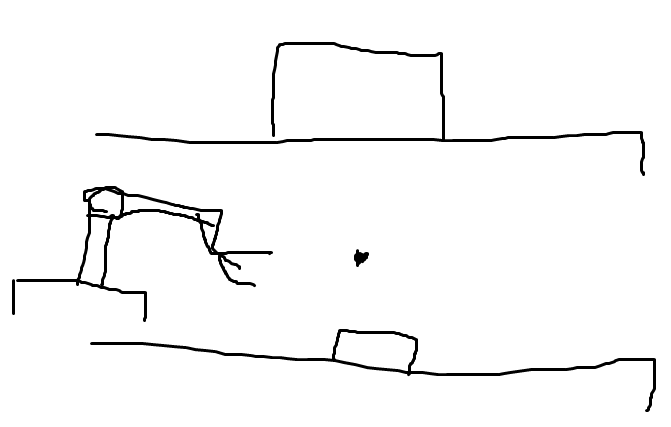
**Date: March 20, 2021**

**1)** Program the robot to pick up two blocks (the blocks are different sizes) from fixed positions on either side of a center position, and stack the blocks in the center position. The larger block will always be on one side of the center and the smaller block will always be on the other side of the center position. The smaller block is to be placed on top of the larger block.

\*\*PROGRAM DEFINITIONS (GLOBAL)\*\*

* 1: DEFINE point C\_L //center point for Large block placement at the center point
* 2: DEFINE point C\_S //center point for Small block placement atop Large block
* 3: DEFINE point H // home position 100mm above point C\_L
* 4: DEFINE point L\_H // (High) above center of Large block
* 5: DEFINE point L\_G // grab position for the Large block
* 6: DEFINE point L\_H //100mm (High) above center of Large block
* 7: DEFINE point L\_G // grab position for the Large block
* 8: DEFINE point S\_H // (High) above center of Small block
* 9: DEFINE point S\_G // grab position for the Small block
* 10: DEFINE line 1: open Gripper //output line to open gripper
* 11: DEFINE line 2: close Gripper //output line to close gripper
* 12: DEFINE line 3: move Conveyors //moves the conveyors of both large and small blocks in order to obtain a new set of blocks

\*\*PROGRAM SUBROUTINES (MACROS/BRANCHES/FUNCTIONS)\*\*

* 13: BRANCH GET\_LARGE
* 14: MOVE L\_H //move to Large home position
* 15: MOVE L\_G //move to Large grab position
* 16: SIGNAL 2 //close gripper
* 17: MOVE L\_H //move back to Large home position
* 18: MOVE H //move to home location
* 19: MOVE C\_L //move to center location
* 20: SIGNAL 1 //open gripper; release Large block
* 21: END GET\_LARGE
* 22: BRANCH GET\_SMALL
* 23: MOVE S\_H //move to Small home position
* 24: MOVE S\_G //move to Small grab position
* 25: SIGNAL 2 //close gripper
* 26: MOVE S\_H //move back to Small home position
* 27: MOVE H //move to home location
* 28: MOVE C\_S //move to center location for Small block
* 29: SIGNAL 1 //open gripper; release Small block
* 30: END GET\_SMALL

\*\*MAIN FUNCTION\*\*

* 31: MOVE H //start at home position
* 32: SIGNAL 1 //open gripper
* 33: GET\_LARGE //subroutine to get large block and stack it at center
* 34: MOVE H
* 35: GET\_SMALL //subroutine; get small block and stack it at center atop large one
* MOVE H
* 36: SIGNAL 3 //move conveyors to bring forward the next blocks
* 37: GO TO 31 //repeat the main function
* 38: END PROGRAM

**2)** A robot is to be programmed to unload parts from one pallet and load them onto another pallet. The parts are located on the unload pallet (pallet 1) in a 3 by 4 pattern in known fixed positions, 40 mm apart in both directions. The two directions of the pallet are assumed to be parallel to the x and y world coordinate axes of the robot. The parts are to be placed on the load pallet (pallet 2) in a 2 by 6 pattern, 40 mm apart in both directions. The two directions of the pallet are again assumed to be parallel to the x and y world coordinate axes of the robot. Make a sketch of the workstation setup before you begin programming.

|  |  |  |  |
| --- | --- | --- | --- |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart | 40mm apart | 40mm apart |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart | 40mm apart | 40mm apart |

**Pallet 2**

**Pallet 1**

40mm

0mm

80mm

40mm

0mm

**Y**

0mm 40mm 80mm 160mm 200mm 240mm

0mm 40mm 80mm 160mm

**X**

\*\*PROGRAM DEFINITIONS (GLOBAL)\*\*

1. DEFINE point P1\_CORNER = [JOINT COORD] //define center for bottom-left corner of pallet 1
2. DEFINE point P2\_CORNER = [JOINT COORD] //define center for bottom-left corner of pallet 2
3. DEFINE INTEGER X\_P1 //x-coordinate value
4. DEFINE INTEGER Y\_P1 //y-coordinate value
5. DEFINE INTEGER X\_P2 //x-coordinate value
6. DEFINE INTEGER Y\_P2 //y-coordinate value
7. DEFINE PICKUP\_P1 = COORDINATES (X, Y)
8. DEFINE DROP\_P2 = COORDIANTES (X, Y)
9. DEFINE int ROW\_P1
10. DEFINE int COL\_P1
11. DEFINE int ROW\_P2
12. DEFINE int COL\_P2

\*\*PROGRAM SUBROUTINES (MACROS/BRANCHES/FUNCTIONS)\*\*

1. **BRANCH REINITIALIZE\_ROW\_COL**
2. IF COL\_P1 == 3
3. THEN:
   1. COL\_P1 = 0
   2. ROW\_P1 = ROW\_P1 + 1 //after reinitializing a column, it means we need to go to the next row
4. ELSE:
   1. COL\_P1 = COL\_P1 + 1 //only increase columns if it is not out of range
5. END IF
6. IF COL\_P2 == 5
7. THEN:
   1. COL\_P2 = 0
   2. ROW\_P2 = ROW\_P2 + 1 //after reinitializing a column, it means we need to go to the next row
8. ELSE:
   1. COL\_P2 = COL\_P2 + 1 //only increase columns if it is not out of range
9. END IF
10. END REINITIALIZE\_ROW\_COL
11. **BRANCH SET\_COORDINATES**
12. X\_P1 = COL\_P1 \* 40
13. X\_P2 = COL\_P2 \* 40 //multiply the X and Y coordinate values by the spacing == 40mm
14. Y\_P1 = ROW\_P1 \* 40
15. Y\_P2 = ROW\_P2 \* 40
16. END SET\_COORDINATES

\*\*MAIN FUNCTION\*\*

1. OPENI //start with gripper open
2. ROW\_P1 = 0
3. ROW\_P2 = 0 //initialize all rows to 0 at start of program
4. COL\_P1 = -1 //initialize columns to -1 so that they will go into the branch statement and be set to 0
5. COL\_P2 = -1
6. REINITIALIZE\_ROW\_COL //increment rows or columns as necessary, reset columns to 0 if necessary
7. IF (ROW\_P1 == 3 AND ROW\_P2 == 2) GO TO 48 //if rows and columns out of scope; end program
8. SET\_COORDINATES //set coordinate values based on ROW and COL and 40mm spacing
9. PICKUP\_P1 = P1\_CORNER + <X\_P1, Y\_P1> //offset from pickup corner on pallet 1
10. APPROS PICKUP\_P1 50 //approach pickup point from 50mm above
11. MOVE PICKUP\_P1
12. CLOSEI
13. DEPART 50
14. DROP\_P2 = P2\_CORNER + <X\_P2, Y\_P2> //offset from drop-off corner on pallet 2
15. APPROS DROP\_P2 50
16. MOVE DROP\_P2
17. OPENI
18. DEPART 50
19. GO TO 34
20. END PROGRAM

**3)** Write a robot program to pick parts off a conveyor and load them into a pallet that is about 12 in. from the pickup point. A mechanical stop on the conveyor is used to locate the parts in a known position for the pickup. The parts are to be arranged in a 3 by 4 pattern, 40 mm apart in both directions. The two directions of the pallet are assumed to be parallel to the x and y world coordinate axes of the robot, respectively.

|  |  |  |  |
| --- | --- | --- | --- |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart |
| 40mm apart | 40mm apart | 40mm apart | 40mm apart |

\*\*PROGRAM DEFINITIONS (GLOBAL)\*\*

**12 in.**

CONVEYOR

**X**

**Y**

1. DEFINE point PALLET\_CORNER = [JOINT COORD] //define center for bottom-left corner of pallet
2. DEFINE point PART\_LOCATION = [JOINT COORD] //define center of part location on the conveyor
3. DEFINE INTEGER X\_PALLET //x-coordinate value
4. DEFINE INTEGER Y\_PALLET //y-coordinate value
5. DEFINE DROP\_PALLET = COORDIANTES (X, Y)
6. DEFINE int ROW\_PALLET
7. DEFINE int COL\_PALLET

\*\*PROGRAM SUBROUTINES (MACROS/BRANCHES/FUNCTIONS)\*\*

1. **BRANCH REINITIALIZE\_ROW\_COL**
2. IF COL\_PALLET == 3
3. THEN:
   1. COL\_P1 = 0
   2. ROW\_P1 = ROW\_P1 + 1 //after reinitializing a column, it means we need to go to the next row
4. ELSE:
   1. COL\_P1 = COL\_P1 + 1 //only increase columns if it is not out of range
5. END IF
6. END REINITIALIZE\_ROW\_COL
7. **BRANCH SET\_COORDINATES**
8. X\_PALLET = COL\_PALLET \* 40
9. Y\_PALLET = ROW\_PALLET \* 40 //multiply the X and Y coordinate values by the spacing == 40mm
10. END SET\_COORDINATES

\*\*MAIN FUNCTION\*\*

1. OPENI //start with gripper open
2. ROW\_PALLET = 0 //initialize all rows to 0 at start of program
3. COL\_PALLET = -1 //initialize columns to -1 so that they will go into the branch statement and be set to 0
4. REINITIALIZE\_ROW\_COL //increment rows or columns as necessary, reset columns to 0 if necessary
5. IF (ROW\_P1 == 3) GO TO 34 //if rows and columns out of scope; end program
6. SET\_COORDINATES //set coordinate values based on ROW and COL and 40mm spacing
7. APPROS PART\_LOCATION 50 //approach pickup point from 50mm above
8. MOVE PART\_LOCATION
9. CLOSEI //close gripper
10. DEPART 50
11. DROP\_PALLET = PALLET\_CORNER + <X\_PALLET, Y\_PALLET> //offset from drop-off corner on pallet 2
12. APPROS DROP\_PALLET 50
13. MOVE DROP\_PALLET
14. OPENI
15. DEPART 50
16. GO TO 21
17. END PROGRAM

**4)** Write a simple program to move the robot through a path that consists of 15 points along the line Y= 0.5X.

\*\*PROGRAM DEFINITIONS (GLOBAL)\*\*

1. DEFINE INTEGER X
2. DEFINE INTEGER Y
3. DEFINE INTEGER COUNTER
4. DEFINE point INITIAL = [JOINT COORD] //use this to record an initial starting point
5. DEFINE point LINEAR = COORDINATES (X, Y) //use this to store the XY coordinates of a point

\*\*PROGRAM SUBROUTINES (MACROS/BRANCHES/FUNCTIONS)\*\*

1. BRANCH SET\_XY
2. X = X + 1
3. Y = 0.5 \* X //set Y based on X
4. END SET\_XY

\*\*MAIN FUNCTION\*\*

1. COUNTER = 0
2. X = -1 //initialize X at -1 since first branch call will set it to 0
3. SET\_XY //this branch will initialize Y
4. LINEAR = INITIAL + <X, Y> //set new position for LINEAR variable
5. MOVES LINEAR //move in a straight line to location stored in LINEAR
6. COUNTER = COUNTER + 1 //now increment counter
7. IF (COUNTER < 15) GO TO 12 //set X and Y to new values if COUNTER < 15
8. END PROGRAM

**5)** An industrial robot performs a machine loading and unloading operation. A PLC is used as the robot cell controller. The cell operates as follows:

* -(1) a human worker places a work part into a nest,
* -(2) the robot reaches over and picks up the part and places it into an induction heating coil,
* -(3) a time of 10 seconds is allowed for the heating operation, and
* -(4) the robot reaches in and retrieves the part and places it on an outgoing conveyor.
* A limit switch I1 (normally open) will be used in the nest to indicate part presence in step (1).
* Similarly the presence of a part on the induction heating coil in step (2) and the outgoing conveyor in step (4) will be detected by I2 and I3, respectively.
* Output contact O1 will be used to signal the robot to execute step (2) of the work cycle. This is an output contact for the PLC, but an input interlock for the robot controller.
* Timer T1 will be used to provide the 10 second delay in step (3).
* Output contact O2 will be used to signal the robot to execute step (4).

Construct the ladder logic diagram for the system.

* I1 = detect part presence in the nest
* I2 = detect presence of a part in the heating coil
* I3 = detect presence of part on conveyor
* O1 = signal robot to execute pick up part from nest and place into heating coil (This is an output contact for the PLC, but an input interlock for the robot controller.)
* O2 = signal robot to execute retrieve part from heating coil and place it on conveyor
* T1 = 10 second time delay in heating coil from step 3 (wait)

