Active Compliance Object Tracking in Robotics

Mr. Dietrich A. Gehron

Mr. Nathan J. Kury

Mr. Andrew C. Spisak, CTM

Dr. John R. Wright, Jr., CSTM, F.ATMAE

Overview:

-Basic Setup and Networking of Cognex Vision Systems (InSight Software) and Mitsubishi Industrial Robots (RT ToolBox2/MELFABasic)

-Acquiring positional data acquisition of the object in Insight

-MELFABasic
Code required to read the data from Cognex

-Passing the data the interface

-Video demonstrations and tutorials will be shared

Need:

- Machine vision and industrial robots are two modern technologies that are used to automate industrial processes globally.

- Record sales nearly every quarter for industrial robots in the United States.

- Future automation engineers will need to increase their knowledge of both machine vision and industrial robotics to
  solve complex problems in the workplace.

https://www.a3automate.org/robotics-vision-motion-control-industries-set-new-growth-records-2017/

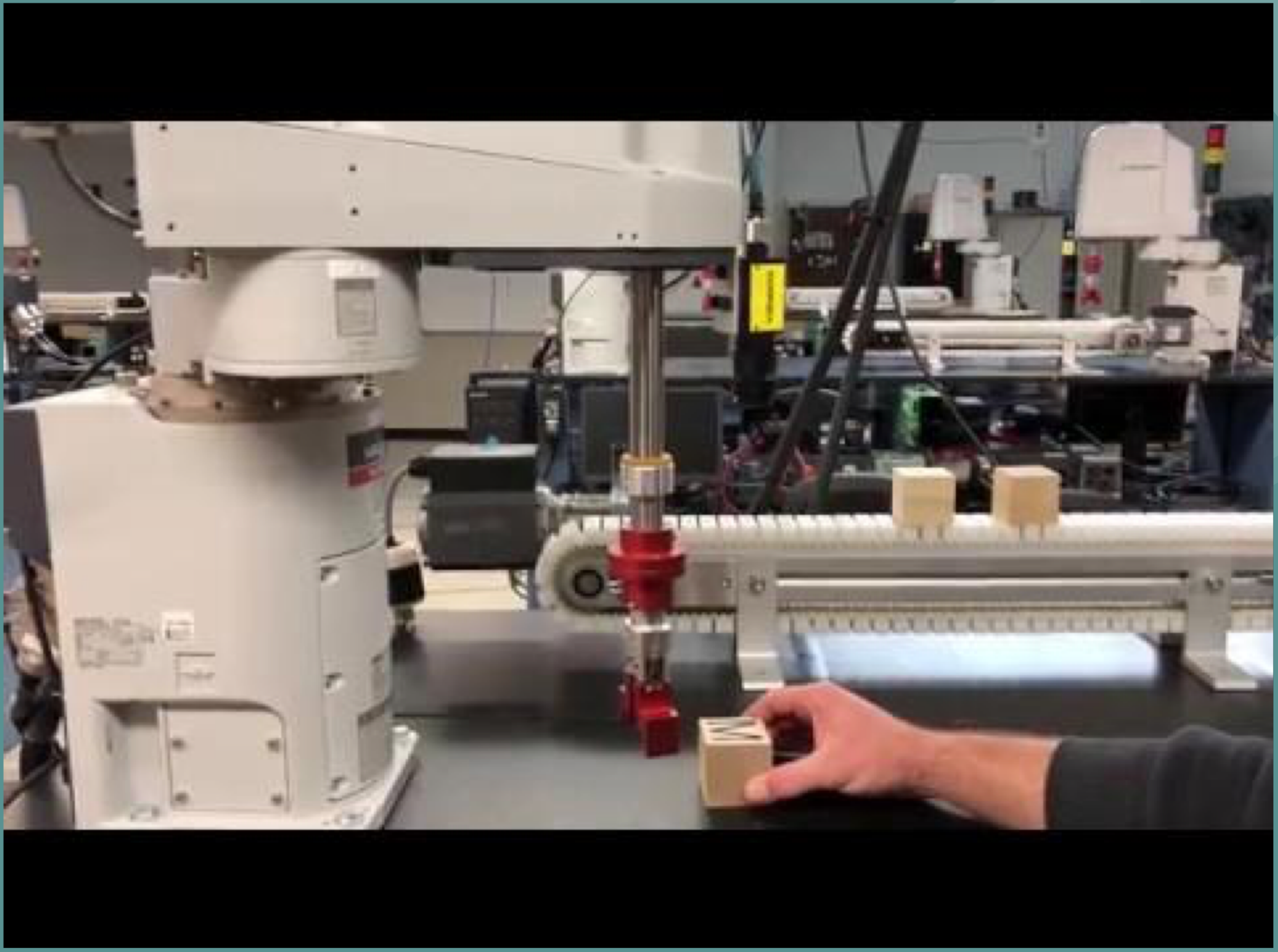
History of Research at Millersville University

2017 - Mr. Nicholas A. Bozzelli, Mr. Michael P. Wiles, Dr. John R. Wright, Mr. Quentin D. Kilgore, and Mr. Kevin L. Wagner developed communication algorithm between Cognex Insight and Mitsubishi Robot

-The future project proposed was real-time tracking with a quarter, we accomplished this with a block.

http://sites.millersville.edu/jwright/Enacting%20Active%20Compliant%20Visual%20Robotic%20Control%20jw.pptx.pdf

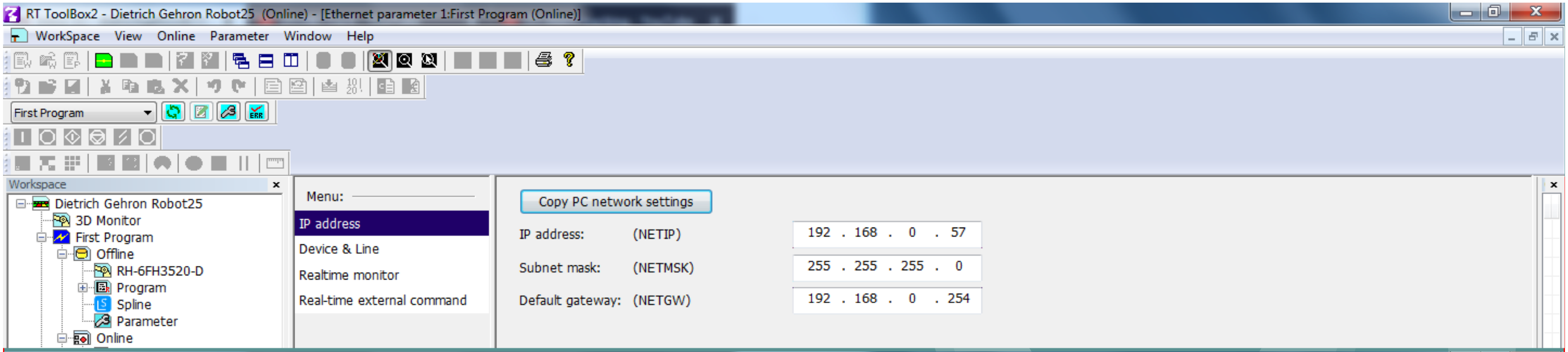
Video Demonstration



https://www.youtube.com/watch?v=faTmwMiJNao&feature=youtu.be

Basic Setup and Networking of Cognex and Mitsubishi

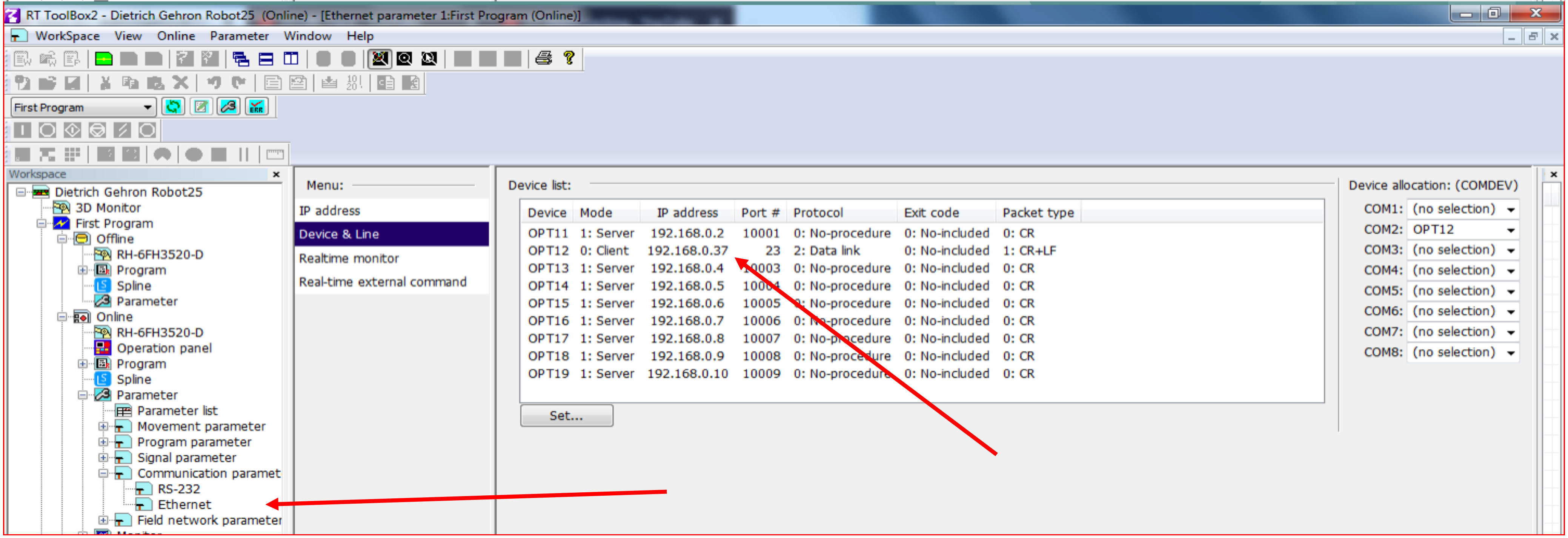
Mr. Nathan J. Kury



IP address: (NETIP) 192 . 168 . 0 . 57

Subnet mask: (NETMSK) 255 . 255 . 255 . 0

Default gateway: (NETGW) 192 . 168 . 0 . 254

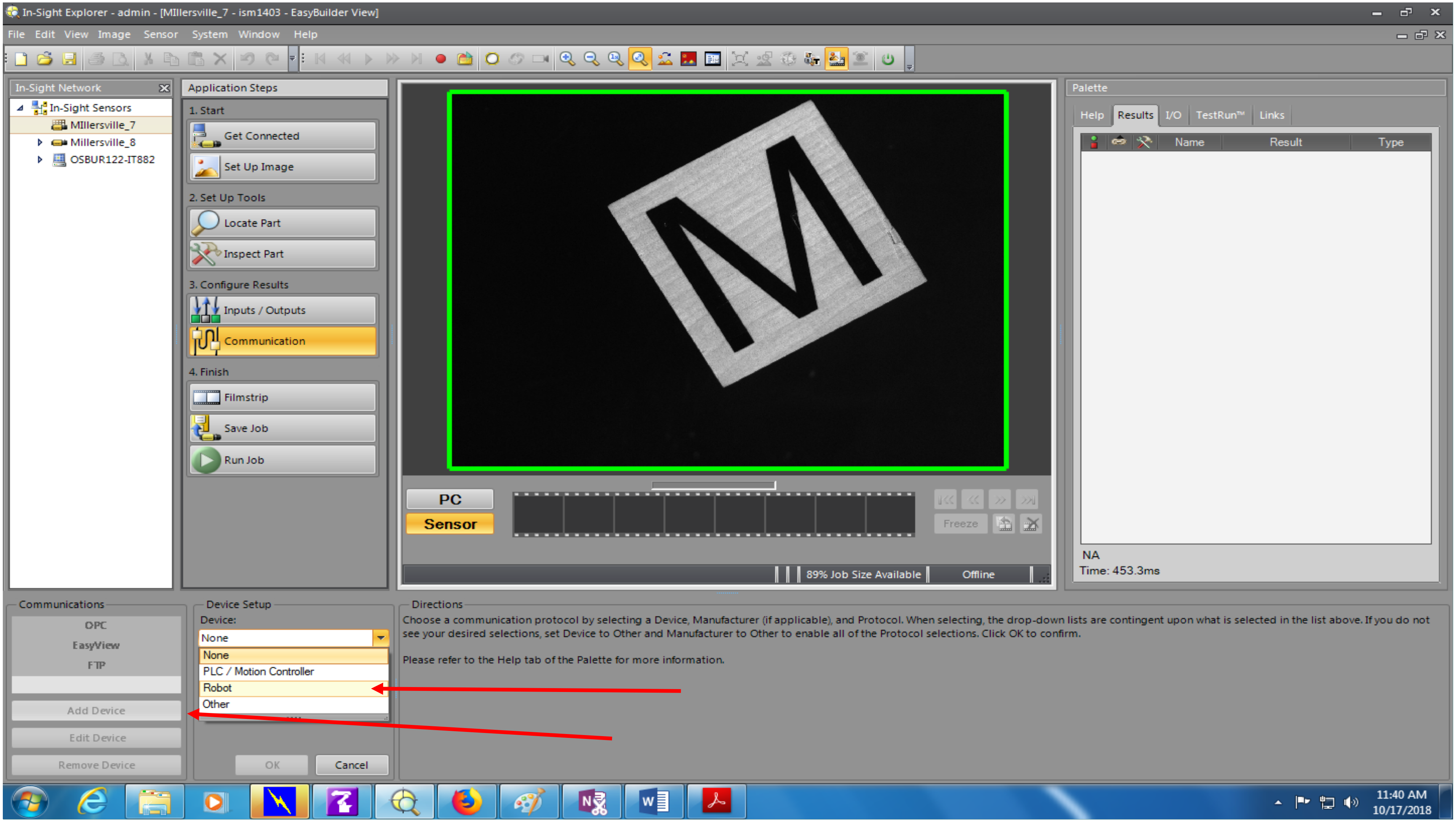


Device list:

Device	Mode	IP address	Port #	Protocol	Exit code	Packet type
OPT11	1: Server	192.168.0.2	10001	0: No-procedure	0: No-included	0: CR
OPT12	0: Client	192.168.0.37	23	2: Data link	0: No-included	1: CR+LF
OPT13	1: Server	192.168.0.4	10003	0: No-procedure	0: No-included	0: CR
OPT14	1: Server	192.168.0.5	10004	0: No-procedure	0: No-included	0: CR
OPT15	1: Server	192.168.0.6	10005	0: No-procedure	0: No-included	0: CR
OPT16	1: Server	192.168.0.7	10006	0: No-procedure	0: No-included	0: CR
OPT17	1: Server	192.168.0.8	10007	0: No-procedure	0: No-included	0: CR
OPT18	1: Server	192.168.0.9	10008	0: No-procedure	0: No-included	0: CR
OPT19	1: Server	192.168.0.10	10009	0: No-procedure	0: No-included	0: CR

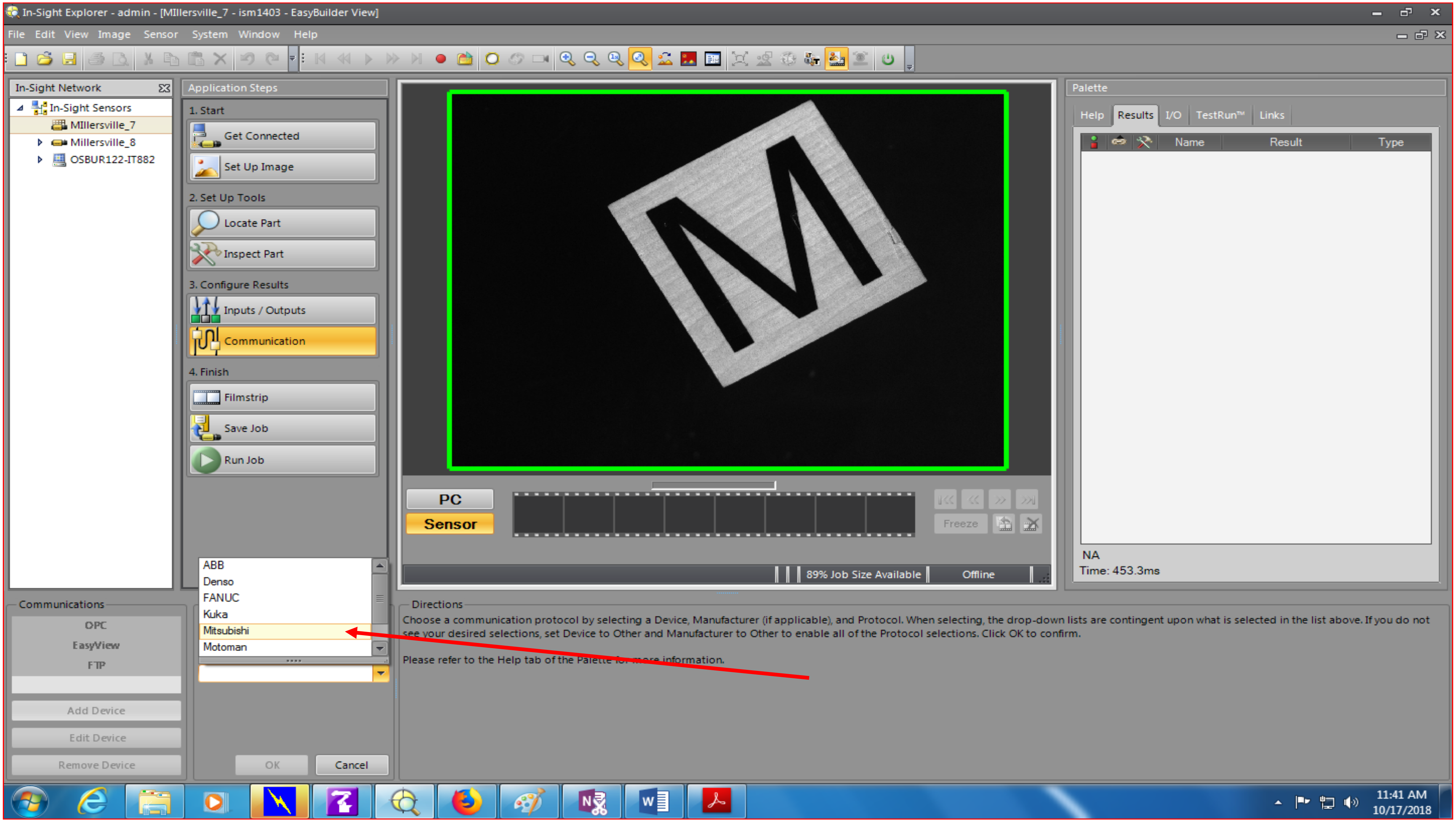
Device allocation: (COMDEV)

COM1:	(no selection)
COM2:	OPT12
COM3:	(no selection)
COM4:	(no selection)
COM5:	(no selection)
COM6:	(no selection)
COM7:	(no selection)
COM8:	(no selection)



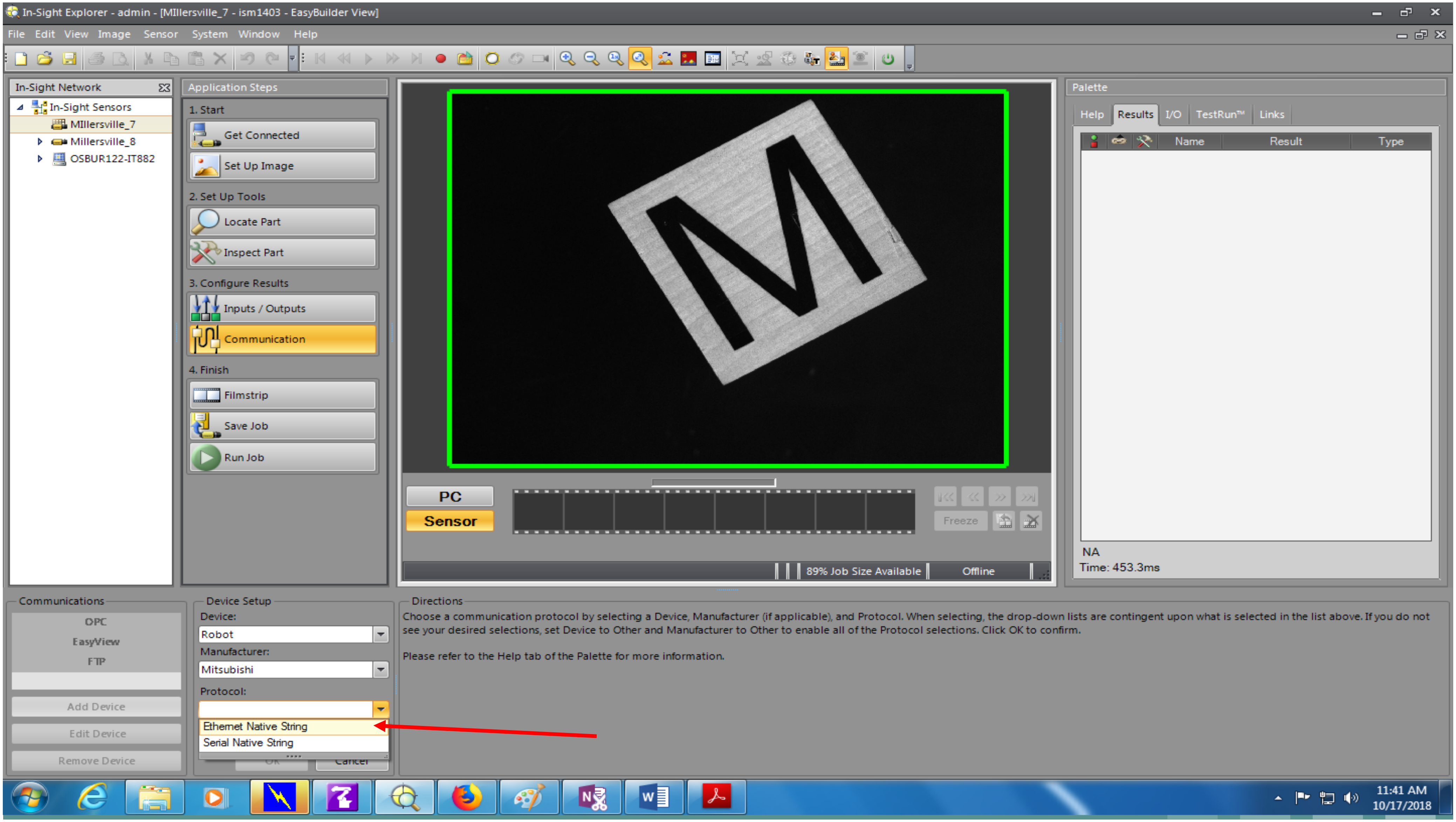
Choose a communication protocol by selecting a Device, Manufacturer (if applicable), and Protocol. When selecting, the drop-down lists are contingent upon what is selected in the list above. If you do not see your desired selections, set Device to Other and Manufacturer to Other to enable all of the Protocol selections. Click OK to confirm.

Please refer to the Help tab of the Palette for more information.



Choose a communication protocol by selecting a Device, Manufacturer (if applicable), and Protocol. When selecting, the drop-down lists are contingent upon what is selected in the list above. If you do not see your desired selections, set Device to Other and Manufacturer to Other to enable all of the Protocol selections. Click OK to confirm.

Please refer to the Help tab of the Palette for more information.



Device: Robot

Manufacturer: Mitsubishi

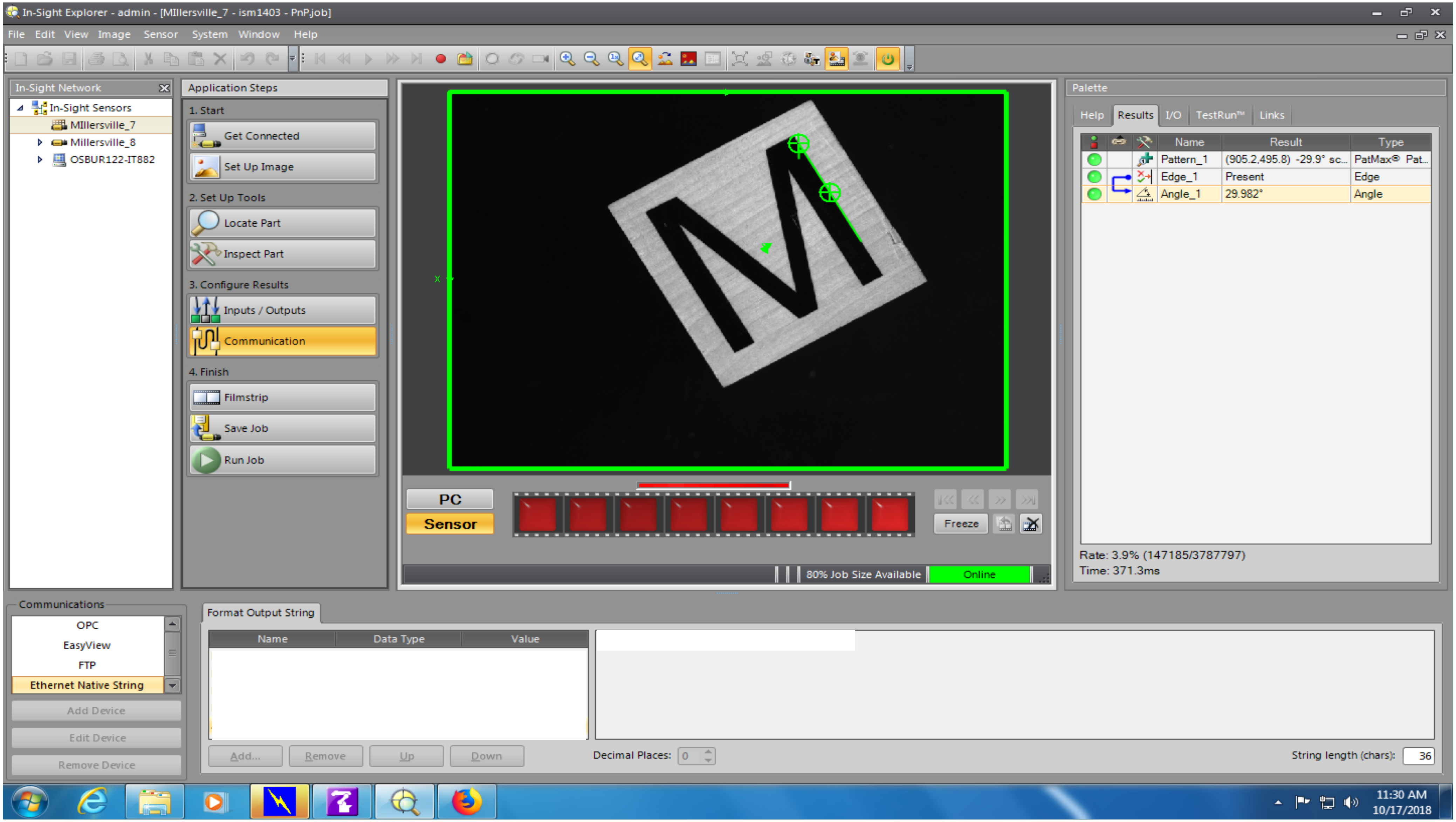
Protocol:

- Ethernet Native String
- Serial Native String

Directions

Choose a communication protocol by selecting a Device, Manufacturer (if applicable), and Protocol. When selecting, the drop-down lists are contingent upon what is selected in the list above. If you do not see your desired selections, set Device to Other and Manufacturer to Other to enable all of the Protocol selections. Click OK to confirm.

Please refer to the Help tab of the Palette for more information.



In-Sight Explorer - admin - [MIllersville_7 - ism1403 - PnP.job]

Name	Result	Type
Pattern_1	(905.2,495.8) -29.9° sc...	PatMax® Pat...
Edge_1	Present	Edge
Angle_1	29.982°	Angle

Rate: 3.9% (147185/3787797)
Time: 371.3ms

80% Job Size Available | Online

Name	Data Type	Value

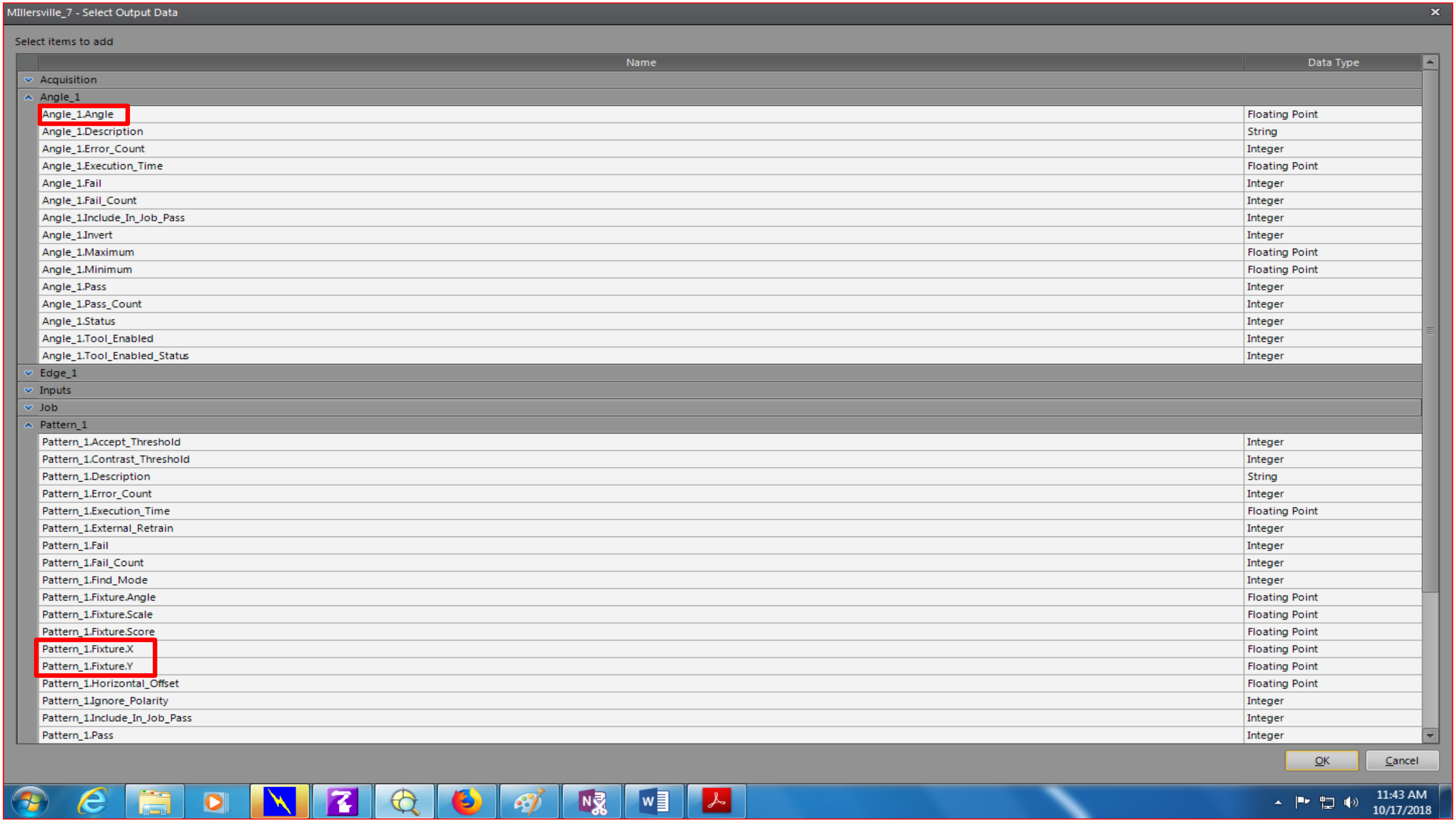
Decimal Places: 0

String length (chars): 36

11:30 AM
10/17/2018

Acquiring Position Data from Cognex

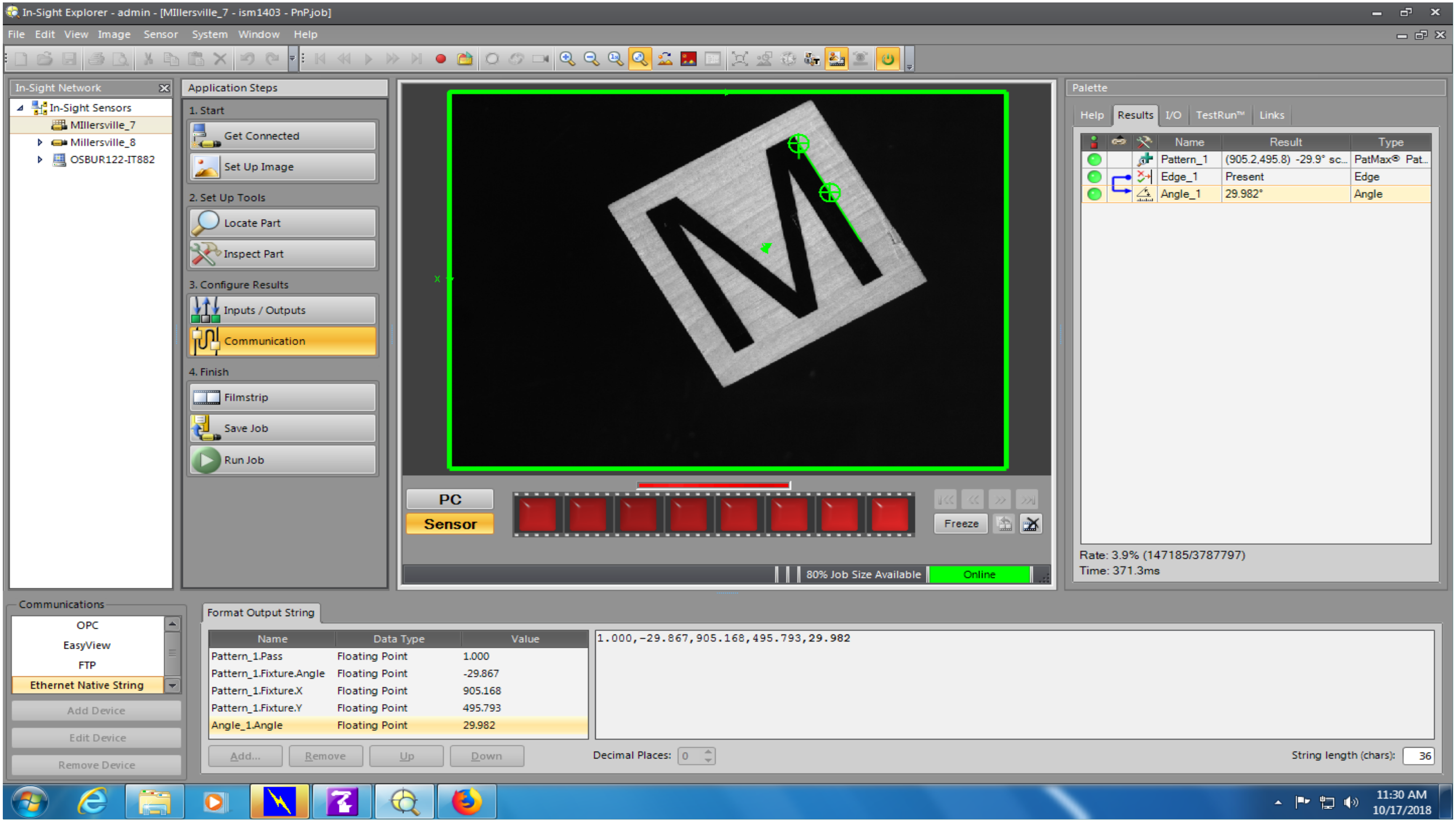
Mr. Nathan J. Kury



MIllersville_7 - Select Output Data

Select items to add

Name	Data Type
Acquisition	
Angle_1	
Angle_1.Angle	Floating Point
Angle_1.Description	String
Angle_1.Error_Count	Integer
Angle_1.Execution_Time	Floating Point
Angle_1.Fail	Integer
Angle_1.Fail_Count	Integer
Angle_1.Include_In_Job_Pass	Integer
Angle_1.Invert	Integer
Angle_1.Maximum	Floating Point
Angle_1.Minimum	Floating Point
Angle_1.Pass	Integer
Angle_1.Pass_Count	Integer
Angle_1.Status	Integer
Angle_1.Tool_Enabled	Integer
Angle_1.Tool_Enabled_Status	Integer
Edge_1	
Inputs	
Job	
Pattern_1	
Pattern_1.Accept_Threshold	Integer
Pattern_1.Contrast_Threshold	Integer
Pattern_1.Description	String
Pattern_1.Error_Count	Integer
Pattern_1.Execution_Time	Floating Point
Pattern_1.External_Retrain	Integer
Pattern_1.Fail	Integer
Pattern_1.Fail_Count	Integer
Pattern_1.Find_Mode	Integer
Pattern_1.Fixture.Angle	Floating Point
Pattern_1.Fixture.Scale	Floating Point
Pattern_1.Fixture.Score	Floating Point
Pattern_1.Fixture.X	Floating Point
Pattern_1.Fixture.Y	Floating Point
Pattern_1.Horizontal_Offset	Floating Point
Pattern_1.Ignore_Polarity	Integer
Pattern_1.Include_In_Job_Pass	Integer
Pattern_1.Pass	Integer



In-Sight Explorer - admin - [MIllersville_7 - ism1403 - PnP.job]

			Name	Result	Type
●			Pattern_1	(905.2,495.8) -29.9° sc...	PatMax® Pat...
●			Edge_1	Present	Edge
●			Angle_1	29.982°	Angle

Rate: 3.9% (147185/3787797)
Time: 371.3ms

80% Job Size Available Online

Name	Data Type	Value
Pattern_1.Pass	Floating Point	1.000
Pattern_1.Fixture.Angle	Floating Point	-29.867
Pattern_1.Fixture.X	Floating Point	905.168
Pattern_1.Fixture.Y	Floating Point	495.793
Angle_1.Angle	Floating Point	29.982

```
1.000,-29.867,905.168,495.793,29.982
```

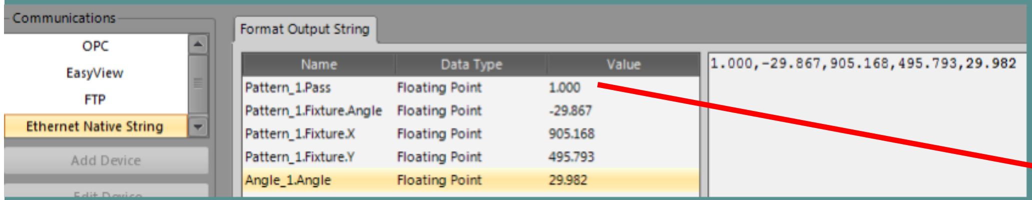
Decimal Places: 0

String length (chars): 36

11:30 AM
10/17/2018

Passing the Data from Cognex to Mitsubishi Robot

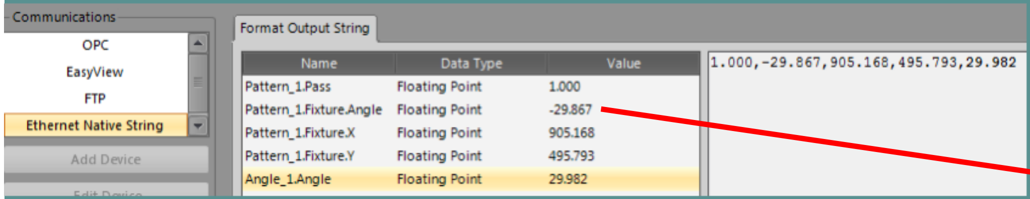
Mr. Nathan J. Kury



Name	Data Type	Value
Pattern_1.Pass	Floating Point	1.000
Pattern_1.Fixture.Angle	Floating Point	-29.867
Pattern_1.Fixture.X	Floating Point	905.168
Pattern_1.Fixture.Y	Floating Point	495.793
Angle_1.Angle	Floating Point	29.982

```
1.000,-29.867,905.168,495.793,29.982
```

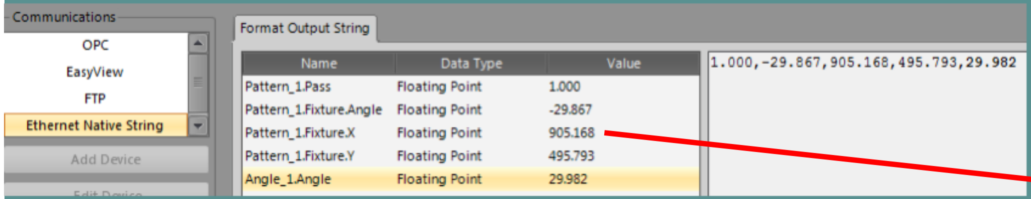
```
'//////////////////Connect Cognex//////////////////////
*ConnectCognex
If M_NvOpen(1) <> 1 Then
Dly 0.1
NVOpen "COM2:" As #1
EndIf
Wait M_NvOpen(1)=1
NVRun #1, "PnP"
Return
'//////////////////END Connect Cognex////////////////////
'
'//////////////////Read Values//////////////////////
*ReadValues
M2 = 0
While M2 <> 1
EBRead #1, ,M2, MNUM,MY,MX,A1
WEnd
Return
'///////////////////END Read Values//////////////////////
```



Name	Data Type	Value
Pattern_1.Pass	Floating Point	1.000
Pattern_1.Fixture.Angle	Floating Point	-29.867
Pattern_1.Fixture.X	Floating Point	905.168
Pattern_1.Fixture.Y	Floating Point	495.793
Angle_1.Angle	Floating Point	29.982

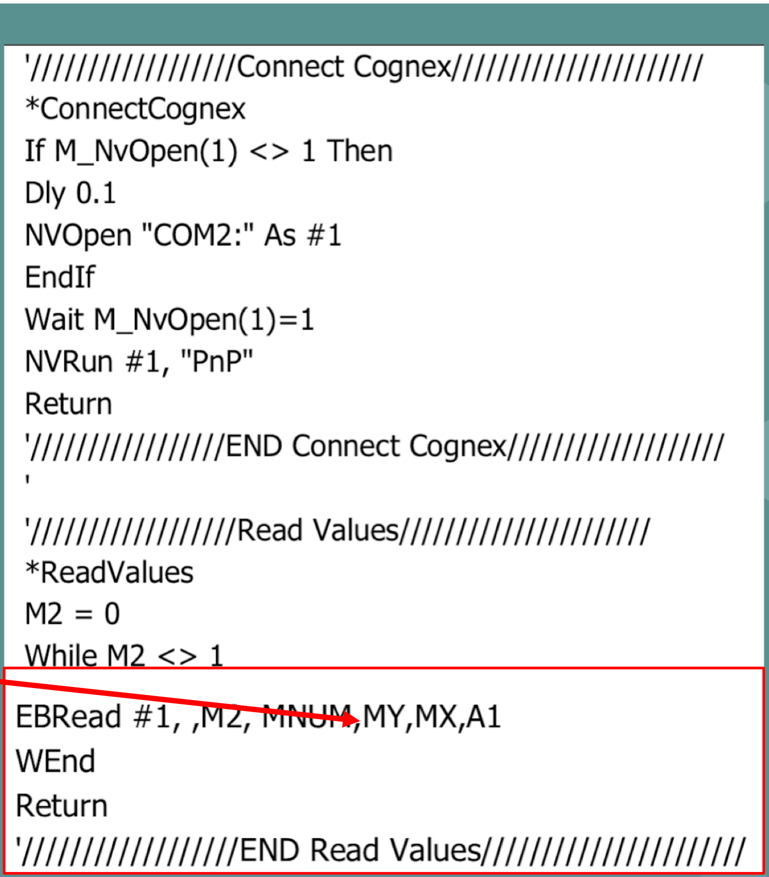
```
1.000,-29.867,905.168,495.793,29.982
```

```
'/////////////////Connect Cognex////////////////////
*ConnectCognex
If M_NvOpen(1) <> 1 Then
Dly 0.1
NVOpen "COM2:" As #1
EndIf
Wait M_NvOpen(1)=1
NVRun #1, "PnP"
Return
'////////////////END Connect Cognex/////////////////
'
'/////////////////Read Values////////////////////
*ReadValues
M2 = 0
While M2 <> 1
EBRead #1, ,M2, MNUM,MY,MX,A1
WEnd
Return
'/////////////////END Read Values////////////////////
```

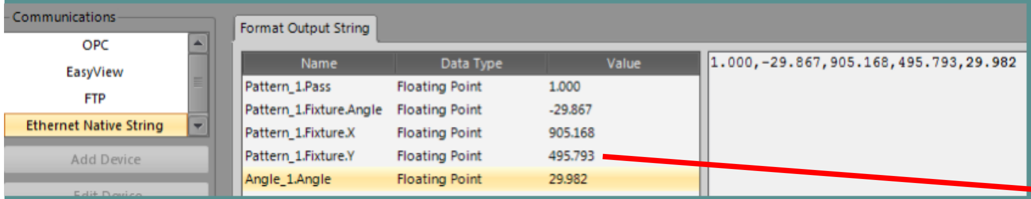


Name	Data Type	Value
Pattern_1.Pass	Floating Point	1.000
Pattern_1.Fixture.Angle	Floating Point	-29.867
Pattern_1.Fixture.X	Floating Point	905.168
Pattern_1.Fixture.Y	Floating Point	495.793
Angle_1.Angle	Floating Point	29.982

```
1.000,-29.867,905.168,495.793,29.982
```



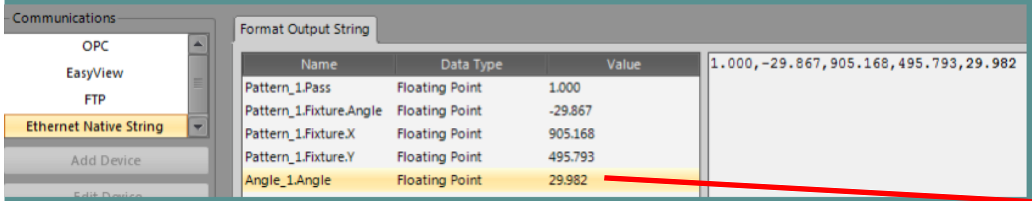
```
'//////////////////Connect Cognex//////////////////////
*ConnectCognex
If M_NvOpen(1) <> 1 Then
Dly 0.1
NVOpen "COM2:" As #1
EndIf
Wait M_NvOpen(1)=1
NVRun #1, "PnP"
Return
'/////////////////END Connect Cognex///////////////////
'

'//////////////////Read Values//////////////////////
*ReadValues
M2 = 0
While M2 <> 1

EBRead #1, ,M2, MNUM,MY,MX,A1
WEnd
Return
'//////////////////END Read Values//////////////////////
```



Name	Data Type	Value
Pattern_1.Pass	Floating Point	1.000
Pattern_1.Fixture.Angle	Floating Point	-29.867
Pattern_1.Fixture.X	Floating Point	905.168
Pattern_1.Fixture.Y	Floating Point	495.793
Angle_1.Angle	Floating Point	29.982

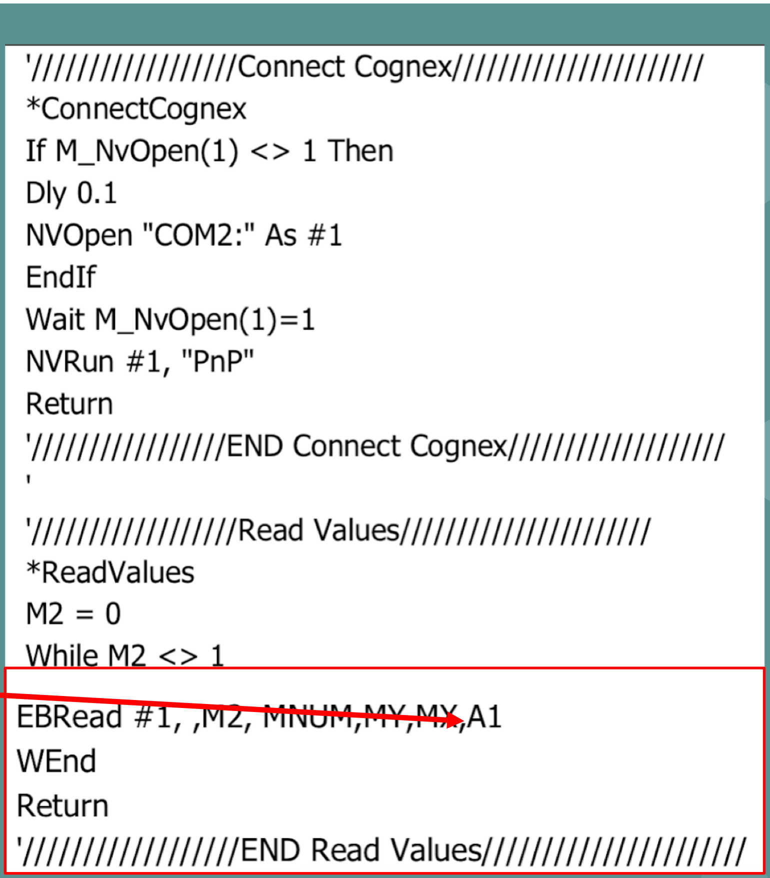
```
1.000,-29.867,905.168,495.793,29.982
```

```
'/////////////////Connect Cognex////////////////////
*ConnectCognex
If M_NvOpen(1) <> 1 Then
Dly 0.1
NVOpen "COM2:" As #1
EndIf
Wait M_NvOpen(1)=1
NVRun #1, "PnP"
Return
'////////////////END Connect Cognex/////////////////
'

'/////////////////Read Values////////////////////
*ReadValues
M2 = 0
While M2 <> 1
EBRead #1, ,M2, MNUM,MY,MX,A1
WEnd
Return
'/////////////////END Read Values////////////////////
```



Name	Data Type	Value
Pattern_1.Pass	Floating Point	1.000
Pattern_1.Fixture.Angle	Floating Point	-29.867
Pattern_1.Fixture.X	Floating Point	905.168
Pattern_1.Fixture.Y	Floating Point	495.793
Angle_1.Angle	Floating Point	29.982

```
1.000,-29.867,905.168,495.793,29.982
```

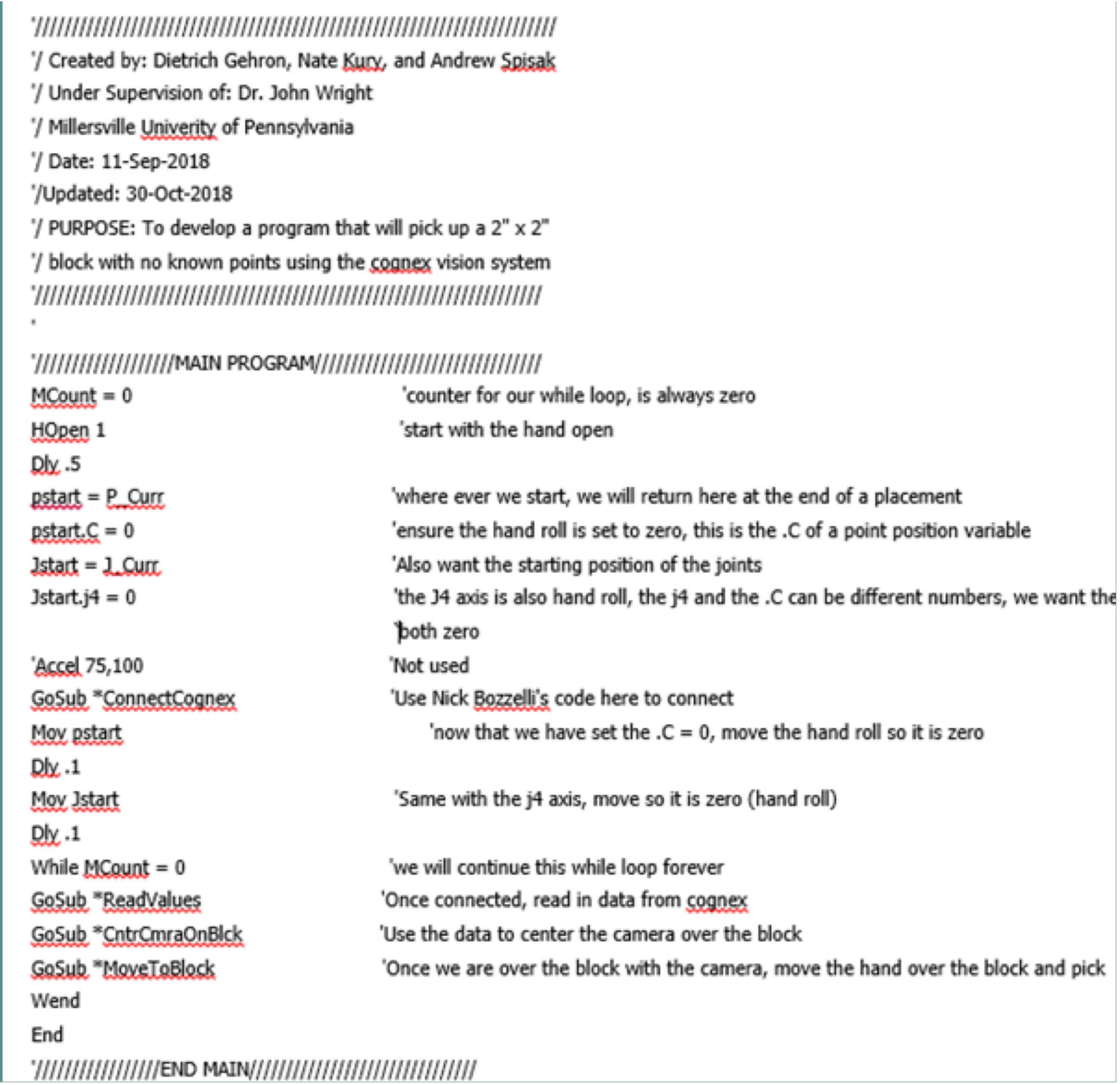


```
'//////////////////Connect Cognex/////////////////////
*ConnectCognex
If M_NvOpen(1) <> 1 Then
Dly 0.1
NVOpen "COM2:" As #1
EndIf
Wait M_NvOpen(1)=1
NVRun #1, "PnP"
Return
'/////////////////END Connect Cognex///////////////////
'

'//////////////////Read Values//////////////////////
*ReadValues
M2 = 0
While M2 <> 1
EBRead #1, ,M2, MNUM,MY,MX,A1
WEnd
Return
'///////////////////END Read Values///////////////////////
```

Melfa-Basic Code

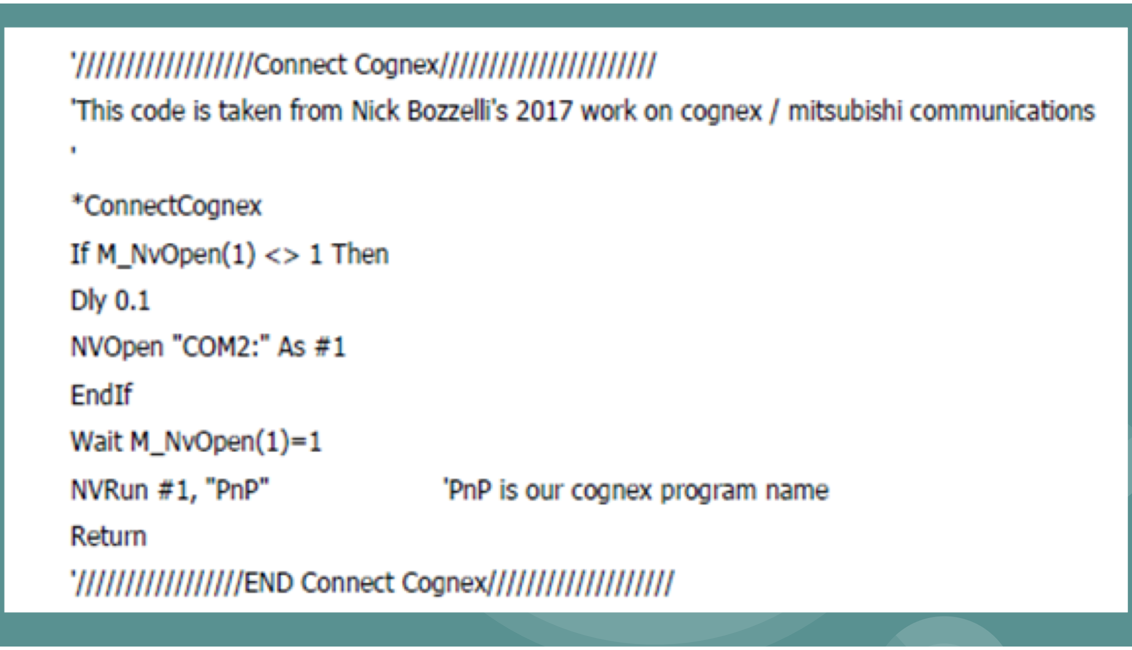
Mr. Dietrich A. Gehron

Main Program



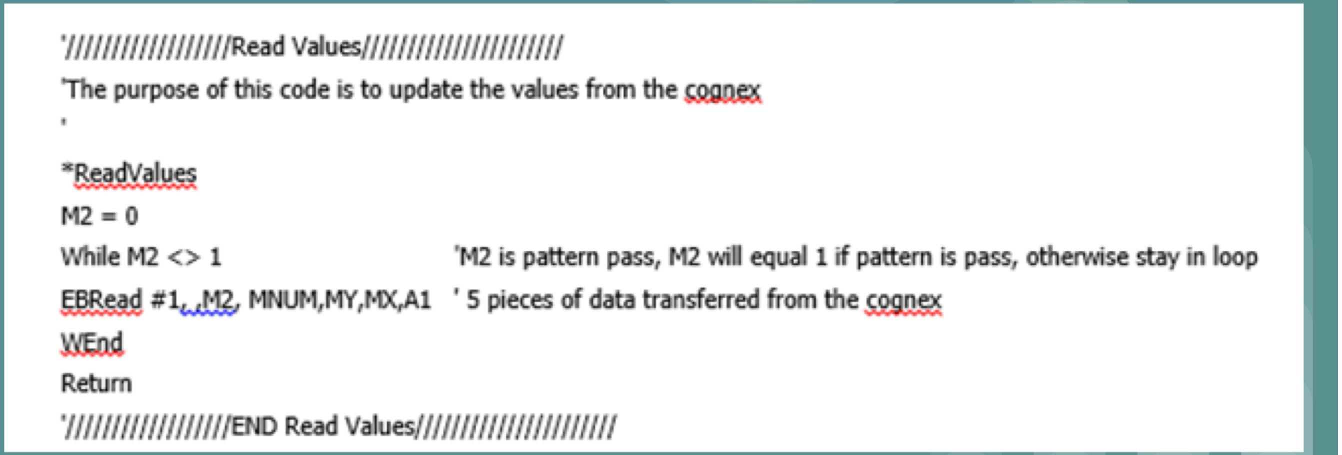
```
'//////////////////////////////////////////////////////////////////////////
'/ Created by: Dietrich Gehron, Nate Kury, and Andrew Spisak
'/ Under Supervision of: Dr. John Wright
'/ Millersville Univerity of Pennsylvania
'/ Date: 11-Sep-2018
'/Updated: 30-Oct-2018
'/ PURPOSE: To develop a program that will pick up a 2" x 2"
'/ block with no known points using the cognex vision system
'////////////////////////////////////////////////////////////////////////
'

'///////////////////MAIN PROGRAM/////////////////////////////////
MCount = 0                              'counter for our while loop, is always zero
HOpen 1                                 'start with the hand open
Dly .5
pstart = P_Curr                         'where ever we start, we will return here at the end of a placement
pstart.C = 0                            'ensure the hand roll is set to zero, this is the .C of a point position variable
Jstart = J_Curr                         'Also want the starting position of the joints
Jstart.j4 = 0                           'the J4 axis is also hand roll, the j4 and the .C can be different numbers, we want the
                                        'both zero
'Accel 75,100                           'Not used
GoSub *ConnectCognex                    'Use Nick Bozzelli's code here to connect
Mov pstart                                  'now that we have set the .C = 0, move the hand roll so it is zero
Dly .1
Mov Jstart                              'Same with the j4 axis, move so it is zero (hand roll)
Dly .1
While MCount = 0                        'we will continue this while loop forever
GoSub *ReadValues                      'Once connected, read in data from cognex
GoSub *CntrCmraOnBlck                  'Use the data to center the camera over the block
GoSub *MoveToBlock                     'Once we are over the block with the camera, move the hand over the block and pick
Wend
End
'/////////////////END MAIN///////////////////////////////
```

Connect Cognex



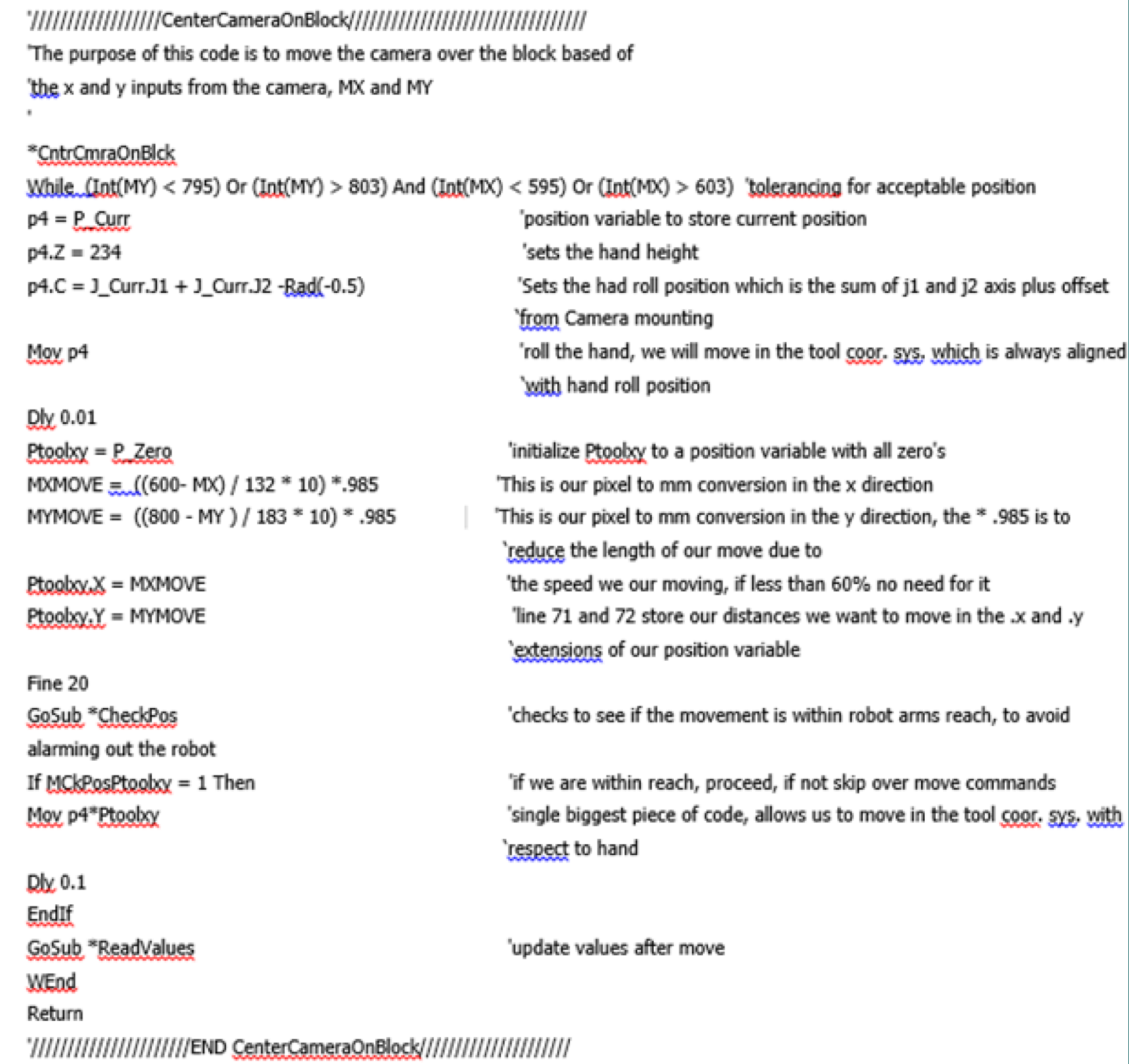
```
'/////////////////Connect Cognex/////////////////////
'This code is taken from Nick Bozzelli's 2017 work on cognex / mitsubishi communications
'
*ConnectCognex
If M_NvOpen(1) <> 1 Then
Dly 0.1
NVOpen "COM2:" As #1
EndIf
Wait M_NvOpen(1)=1
NVRun #1, "PnP"                    'PnP is our cognex program name
Return
'////////////////END Connect Cognex//////////////////
```

Read Values



```
'//////////////////Read Values//////////////////////
'The purpose of this code is to update the values from the cognex
'
*ReadValues
M2 = 0
While M2 <> 1                             'M2 is pattern pass, M2 will equal 1 if pattern is pass, otherwise stay in loop
EBRead #1, ,M2, MNUM,MY,MX,A1   ' 5 pieces of data transferred from the cognex
WEnd
Return
'//////////////////END Read Values//////////////////////
```

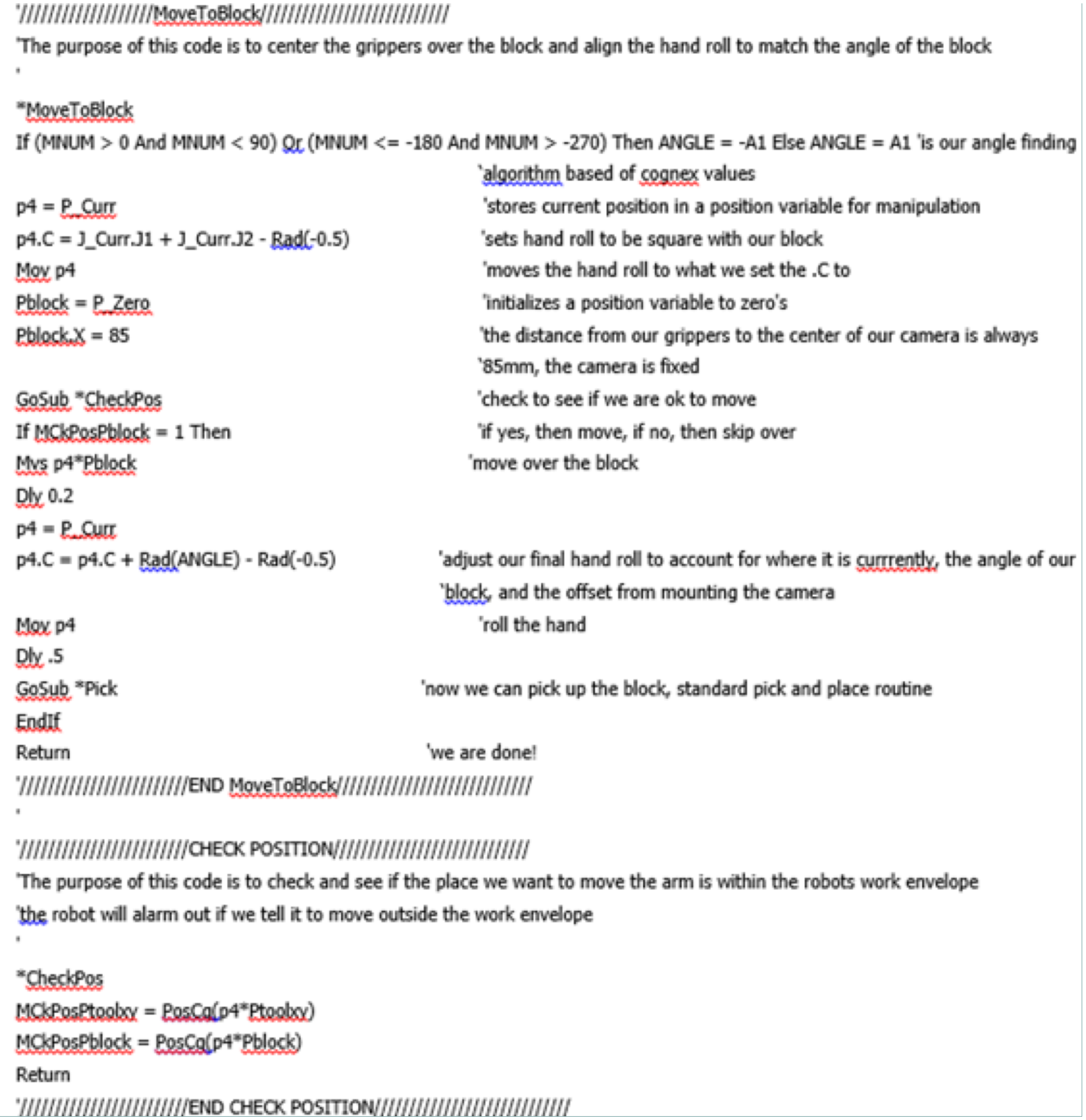
Center Camera on Block



```
'//////////////////CenterCameraOnBlock//////////////////////////////////
'The purpose of this code is to move the camera over the block based of
'the x and y inputs from the camera, MX and MY
'

*CntrCmraOnBlck
While  (Int(MY) < 795) Or (Int(MY) > 803) And (Int(MX) < 595) Or (Int(MX) > 603)  'tolerancing for acceptable position
p4 = P_Curr                                      'position variable to store current position
p4.Z = 234                                       'sets the hand height
p4.C = J_Curr.J1 + J_Curr.J2 -Rad(-0.5)          'Sets the had roll position which is the sum of j1 and j2 axis plus offset
                                                 `from Camera mounting
Mov p4                                           'roll the hand, we will move in the tool coor. sys. which is always aligned
                                                 `with hand roll position

Dly 0.01
Ptoolxy = P_Zero                                'initialize Ptoolxy to a position variable with all zero's
MXMOVE =  ((600- MX) / 132 * 10) *.985         'This is our pixel to mm conversion in the x direction
MYMOVE =  ((800 - MY ) / 183 * 10) * .985      'This is our pixel to mm conversion in the y direction, the * .985 is to
                                                `reduce the length of our move due to
Ptoolxy.X = MXMOVE                              'the speed we our moving, if less than 60% no need for it
Ptoolxy.Y = MYMOVE                              'line 71 and 72 store our distances we want to move in the .x and .y
                                                `extensions of our position variable

Fine 20
GoSub *CheckPos                                 'checks to see if the movement is within robot arms reach, to avoid
alarming out the robot
If MCkPosPtoolxy = 1 Then                       'if we are within reach, proceed, if not skip over move commands
Mov p4*Ptoolxy                                  'single biggest piece of code, allows us to move in the tool coor. sys. with
                                                `respect to hand

Dly 0.1
EndIf
GoSub *ReadValues                               'update values after move
WEnd
Return
'//////////////////////END CenterCameraOnBlock////////////////////
```

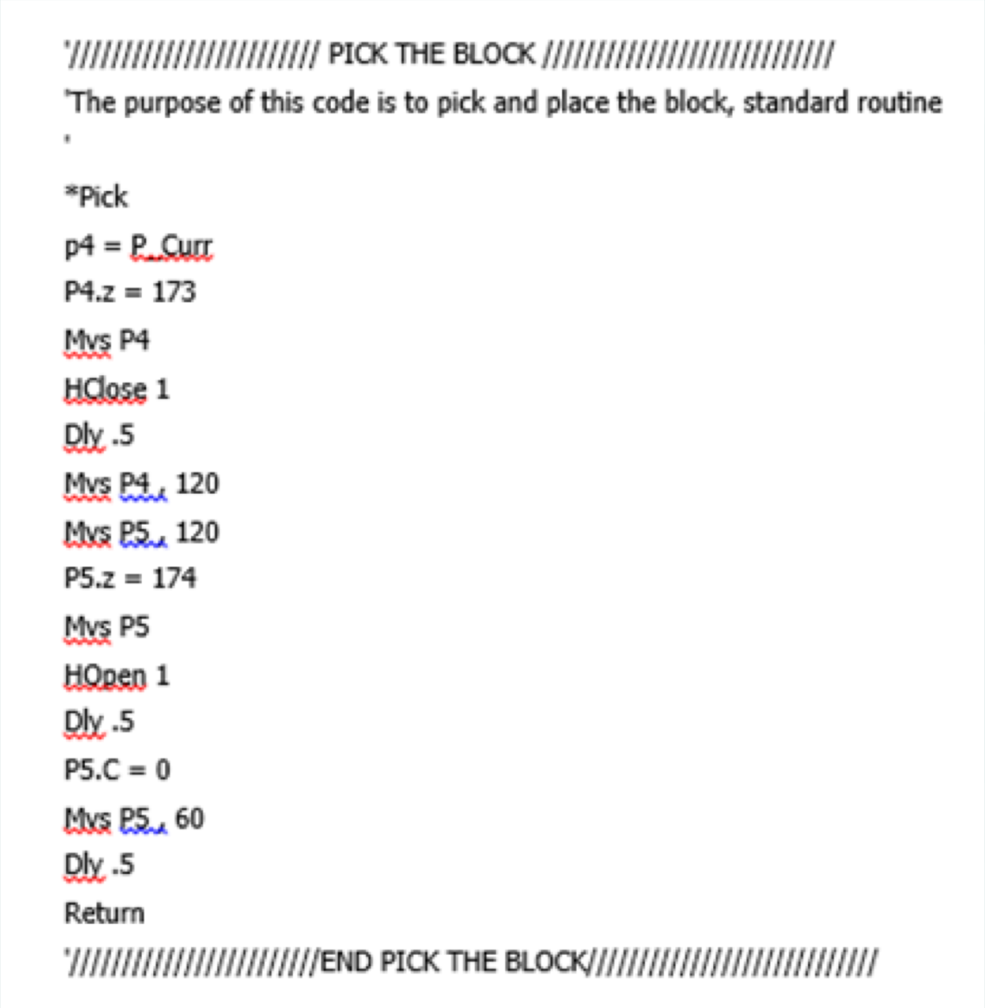
Move to Block

Check Position



```
'///////////////////MoveToBlock//////////////////////////
'The purpose of this code is to center the grippers over the block and align the hand roll to match the angle of the block
'
*MoveToBlock
If (MNUM > 0 And MNUM < 90) Or (MNUM <= -180 And MNUM > -270) Then ANGLE = -A1 Else ANGLE = A1 'is our angle finding
                                                    'algorithm based of cognex values
p4 = P_Curr                                          'stores current position in a position variable for manipulation
p4.C = J_Curr.J1 + J_Curr.J2 - Rad(-0.5)             'sets hand roll to be square with our block
Mov p4                                               'moves the hand roll to what we set the .C to
Pblock = P_Zero                                      'initializes a position variable to zero's
Pblock.X = 85                                        'the distance from our grippers to the center of our camera is always
                                                    '85mm, the camera is fixed
GoSub *CheckPos                                     'check to see if we are ok to move
If MCkPosPblock = 1 Then                            'if yes, then move, if no, then skip over
Mvs p4*Pblock                                      'move over the block
Dly 0.2
p4 = P_Curr
p4.C = p4.C + Rad(ANGLE) - Rad(-0.5)            'adjust our final hand roll to account for where it is currrently, the angle of our
                                                'block, and the offset from mounting the camera
Mov p4                                              'roll the hand
Dly .5
GoSub *Pick                                   'now we can pick up the block, standard pick and place routine
EndIf
Return                                        'we are done!
'//////////////////////////END MoveToBlock///////////////////////////
'
'//////////////////////////CHECK POSITION////////////////////////////
'The purpose of this code is to check and see if the place we want to move the arm is within the robots work envelope
'the robot will alarm out if we tell it to move outside the work envelope
'
*CheckPos
MCkPosPtoolxy = PosCq(p4*Ptoolxy)
MCkPosPblock = PosCq(p4*Pblock)
Return
'//////////////////////////END CHECK POSITION///////////////////////////////
```

Pick the Block



```
'///////////////////////////// PICK THE BLOCK //////////////////////////////////
'The purpose of this code is to pick and place the block, standard routine
'

*Pick
p4 = P_Curr
P4.z = 173
Mvs P4
HClose 1
Dly .5
Mvs P4 , 120
Mvs P5 , 120
P5.z = 174
Mvs P5
HOpen 1
Dly .5
P5.C = 0
Mvs P5 , 60
Dly .5
Return
'/////////////////////////////END PICK THE BLOCK/////////////////////////////////
```

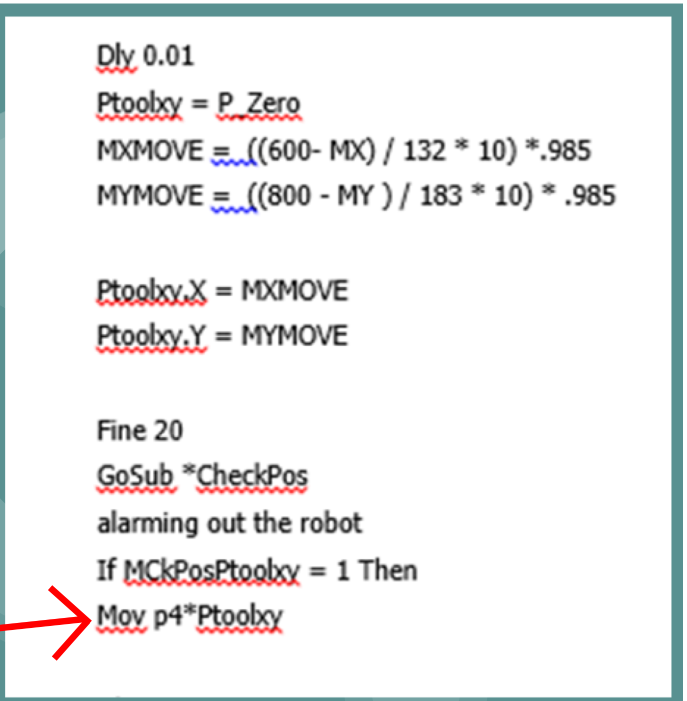
In depth look at code

-P4 is our current position

-the * means move in the Tool coordinate system In addition to the P4 move, which is none, we are already there.

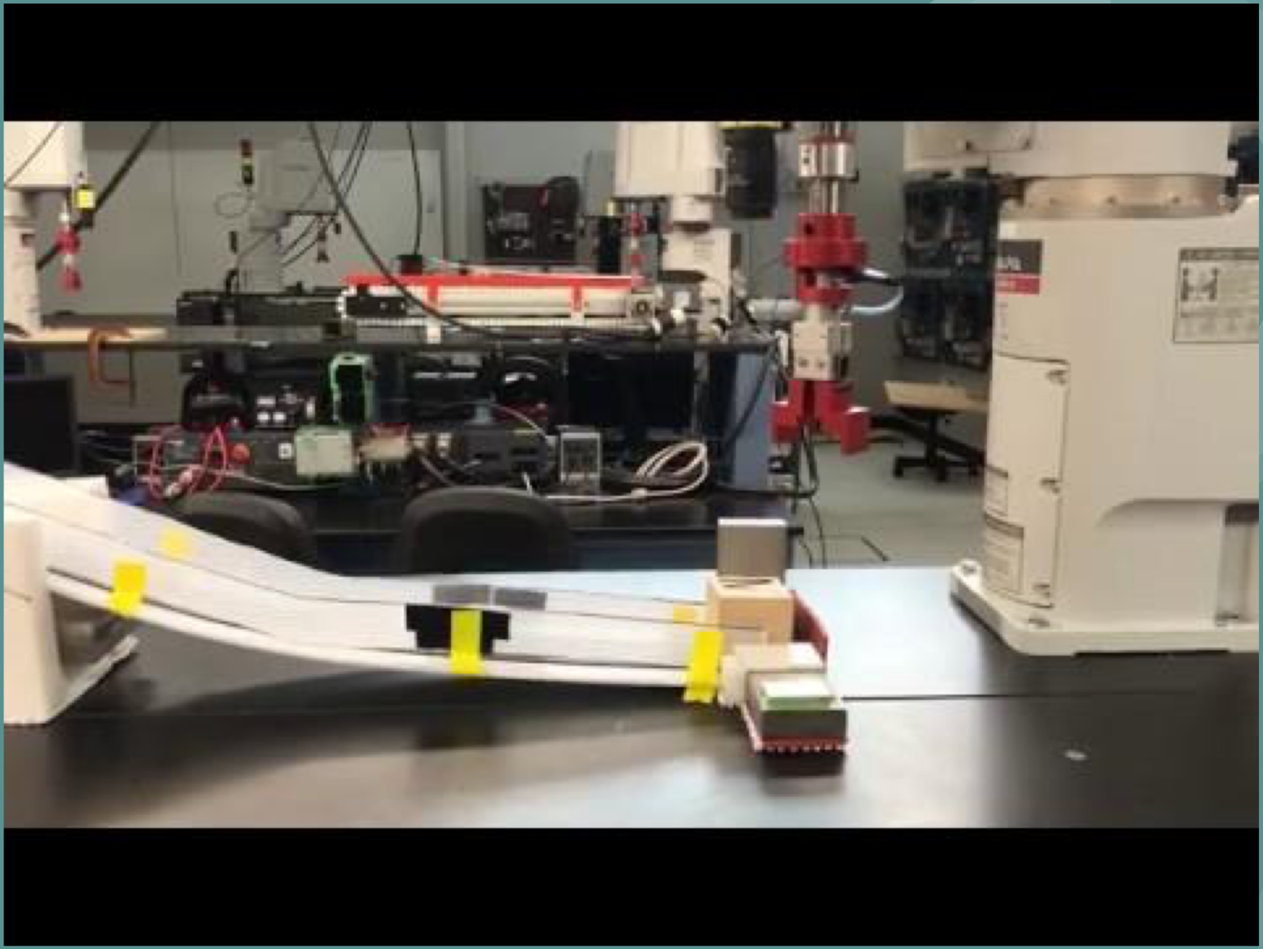
-The x and y distances are stored in the .x and .y extensions of Ptoolxy

-Our "in addition to" move is then only the distances stored in the Ptoolxy in the Tool coor. sys.



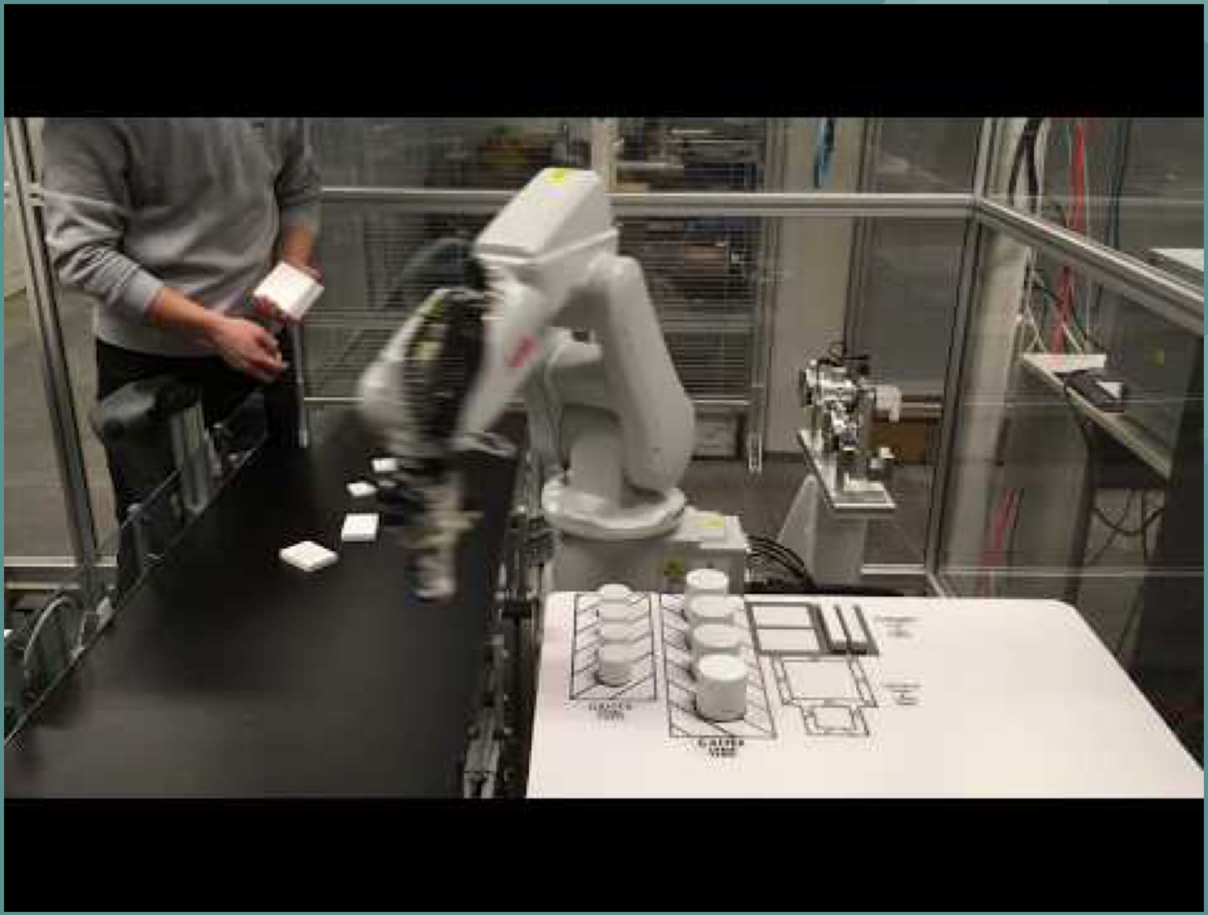
```
Dly 0.01
Ptoolxy = P_Zero
MXMOVE = ((600- MX) / 132 * 10) *.985
MYMOVE = ((800 - MY ) / 183 * 10) * .985

Ptoolxy.X = MXMOVE
Ptoolxy.Y = MYMOVE

Fine 20
GoSub *CheckPos
alarming out the robot
If MCkPosPtoolxy = 1 Then
Mov p4*Ptoolxy
```

Real World Application



https://www.youtube.com/watch?v=G-xHwp9gN60&feature=youtu.be

Future Projects:



https://www.youtube.com/watch?v=7UpR8X4T_vE

Contact Information

All presentations can be found on

- http://sites.millersville.edu/jwright/

Dr. John R. Wright, Jr.

- John.Wright@millersville.edu

Mr. Dietrich A. Gehron

- Dietrichgehron82@gmail.com

Mr. Nathan J. Kury

- N8Kury@Gmail.com

Mr. Andrew C. Spisak

- Spisak.andrew@yahoo.com