Integration of SCARA Robot for Pick and Place Application using PLC

V. Parvathi Priya[1], Jayasuriya Suresh[2]
{vpp.rae@psgtech.ac.in[1], suriyaindu@gmail.com[2]}

Department of Robotics and Automation Engineering, PSG College of Technology, Coimbatore-641 004, India[1,2]

Abstract. In the world of automating factories, automating the process of picking, and placing various items for applications such as sorting, and packaging has become quite popular. Instead of wasting manpower on a repetitive and monotonous task, industries have begun to pass on the task to robots preferably SCARA robots or 6 axis robots. This paper describes how the automation of a pick and place operation is achieved by integrating the robot in an existing production line. The robot used for this application was Epson T6 SCARA robot coupled with Epson's vision system, the CV2 vision system. A Panasonic Programmable Logic Controller was used for other functions such as conveyor movement of trays. A Graphical User Interface based on PyQt5, a python-based application screen was developed. Testing has been done and the robot has been successfully installed onto the production line.

Keywords: SCARA Robot, Pick, Place, Programmable Logic Controller, Automation, Integration.

1 Introduction

Robotics always has been an important part of industrial automation. The year 2020 brought upon the coronavirus pandemic which resulted in massive lockdowns and industries had to shutter down for an extended period of time. The resulting shutdown led to the consideration of robotics as a replacement for human labour as the shutdown caused a severe shortage of labour. Automation reduces the need for dependence on manual labour and enables an increase in productivity and cost savings in the long term.

The pick and place operation of components in the production line has been done manually. With the onset of the new century, automation of the process has started to become popular, but factories preferred manual labour as it was cheaper and easier for the short term. With the shortage of manpower in recent times, replacing manpower with robots for the same process has become popular.

The implementation of SCARA robots and 6 axis robots have become popular for pick and place operations. SCARA robots require less space to install, are compact, easy to install and reliable. The navigation methods for different obstacles the robot might encounter and the options to overcome them during pick and place operations [1].

They also imitate the human arm which offers speed and accuracy for pick and place operations or assembly operations that are between two points on a single plane compared to 6-axis robots. Epson's SCARA robot has been used. A general study of the robot's

ICCAP 2021, December 07-08, Chennai, India

DOI 10.4108/eai.7-12-2021.2314569

specifications, orientation and joints has been done [2]. The robot has been programmed using Epson's RC+7.0. To calculate the position of the Epson SCARA robot and use it to plan an efficient path of travel for the robot to pick and place by finding the relationship between the robot links and the gripper position in the workspace has been studied [3]. The options available in the robot and SPEL+ language to understand the syntax of the language and the various commands available to manipulate the robot to do various tasks has been studied. [8][9][10].

A Panasonic PLC was also used for additional functions required that could not be done by the robot. Panasonic's software suite has been used to program the PLC and Python acts as a core interface to control the robot and PLC. The robot does the identification of 100 components at a time and then moves on to the pick and place workflow. The parts identified by the camera which match the selected component in the system are then picked randomly in an order defined by the camera and placed in a tray in a defined order. This workflow cycle is repeated till all components are picked from the pick tray and a fresh tray is fed.

2 Problem Statement

To integrate a SCARA robot into a pick and place application which is being done manually. For controlling the robot and vision systems of the robot, a graphical user interface has to be developed which should increase the productivity of the assembly line for which the pick and place operation is being done.

Fig. 1 shows the generalised steps of how a pick and place operation is generally executed by a robot programmed to do the same.

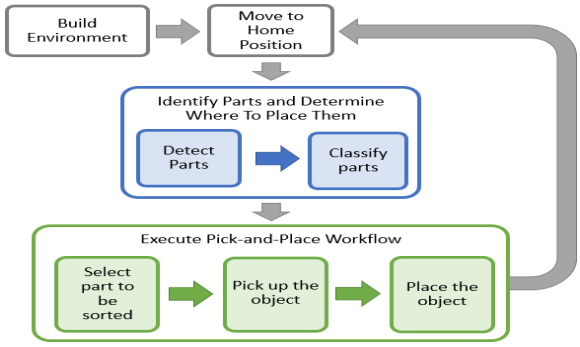

Fig. 1. General Workflow of a Pick and Place Robot

Hardware

The hardware components used are :

- Epson Robot
- Epson Monochrome Camera
- Epson Vision Controller
- Panasonic PLC
- Network Switch

Fig. 2 illustrates the connections are made for the python application running on the pc to the PLC, robot, and vision systems. All devices are assigned with a fixed IP address and communication is generally done via Modbus for passing instructions.

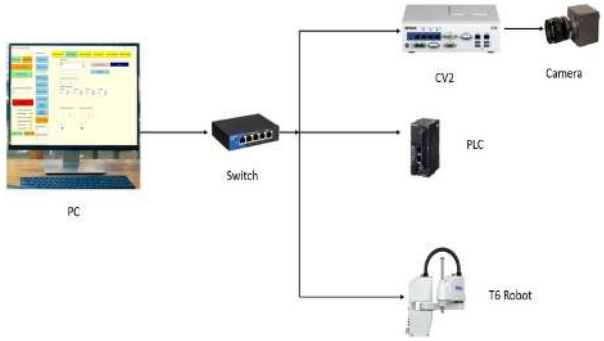

Fig. 2. Hardware Layout

Only for certain cases does Python and the robot communicate directly. All data and control instructions are sent via PLC to the robot for data retention and for the PLC to do certain tasks when a specific instruction is sent and to data log various information that flows within the network such as time running, cycle time and the lifetime components picked and place.

The CV2 vision system and the camera are connected via gigabyte ethernet and connected to the robot via Ethernet. The python application and PLC do not communicate with the CV2 or camera at all. Only the robot and CV2 communicate with each other.

3 Software Used

A. Panasonic Control Fpwin Pro

This is the software used to program Panasonic PLCs. The latest version is v7.0. This has been developed to international standards (IEC 61131-3). It features a wide variety of libraries that are made for easy reusability of ready-made functions and function blocks that are often used.

B. EPSON RC+7.0

This is Epson's software development environment for its series of robots being currently manufactured. This supports many features that make development for Epson's robots simple and intuitive.

The software uses SEPL+ programming language which is based on visual basic and is made for robotic applications. It has many commands for motion, I/O control, pallet operations and many more. Epson provides documentation that is comprehensive and well documented for easy implementation of tasks into a program. There are various categories of syntaxes such as system settings management, executable commands and logic, position and I/O management and robot and motion settings management and executable commands.

Epson Vision guide is a separate module which is tightly integrated into the RC+7.0. The vision guide software is entirely responsible for managing the vision systems installed onto Epson robots. It is simple to use and intuitive.

C. Python Packages

The following Python packages used for the Graphical User Interface:

- Pyqt5
- PyModBusTCP
- Python Subprocess Module
- Python Threading
- PyAutoGui
- Auto Py to Exe

4 Calibration Of The Robot

Robot calibration is done to improve the accuracy of robots, primarily done for industrial robots which have higher repeatability at the cost of lower accuracy. Calibration for the T6 as is done by dividing the Field of View into 9 quadrants and moving the robot into the centre of each quadrant. This enables the camera to know where the robot is picking up.

If the calibration is disturbed by accident, the robot gripper tends to slam into the component, causing chipping of the metal and micro cracks. Calibration is a time-consuming process and is required to be done if the robot is shifted or if the camera is disturbed. Calibration begins with the process of stopping the robot via the Graphical User interface and opening the software from it, which automatically stops the robot and opens RC+7.0.

The calibration wizard acts as an easy interface to calibrate robot, neglecting the need for additional code in the program of the robot.

In the calibration wizard, after defining the name of the calibration and camera, the wizard asks for the mounting of the camera, which in this case is fixing looking downward as shown in Fig. 3.

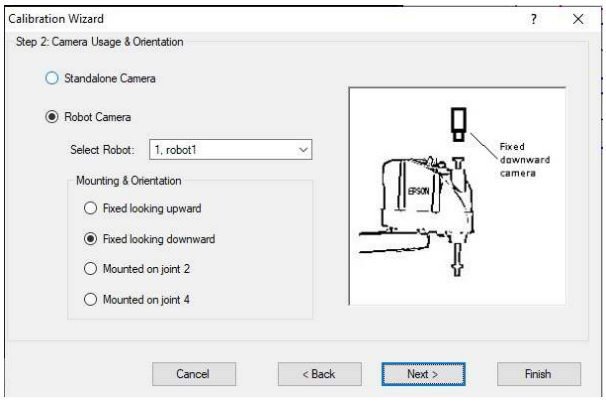

Fig. 3 Camera Position

After the camera position is selected, the wizard asks for the target sequence, which is already defined. A sequence contains the geometric model for the camera to recognise the

component, in this case, to recognize the circles. Then the robot coordinate system is defined, which is zero.

The distortion correction is enabled and the target sequence for distortion correction is selected. Distortion correction is already defined in the sequences manually before calibration is done.

In the jog window, configure the robot is jogged to the nine points and the points are taught accordingly.

As shown in Fig. 4, the calibration chart consists of nine circles with centres marked according to 9 quadrants the cam FOV is divided into. The camera is able to detect the centre of the circles and the robot is jogged to that same point which is marked on the sheet

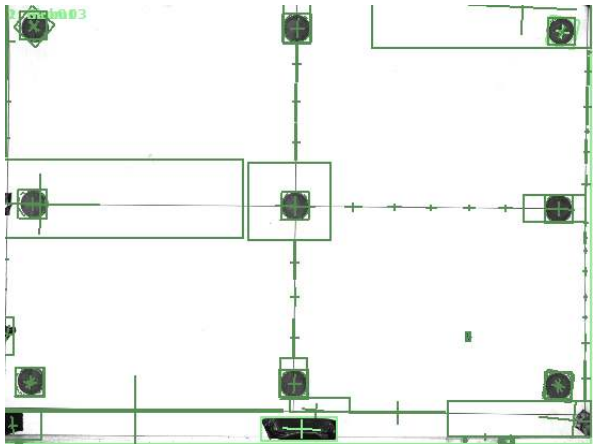

Fig. 4. Calibration Chart

After teaching the points, a summary window pops up and shows a summary of the errors. The max error should be less than 0.3 mm. This also shows the area the camera can cover with its field of view. Having a greater max error with a lower average and vice versa can affect the robot's accuracy as the camera assumes certain points to be covering a larger area due to improper calibration increasing error.

Fig. 5 shows the area the camera can cover and the max error that is desired which is less than 0.3 mm.

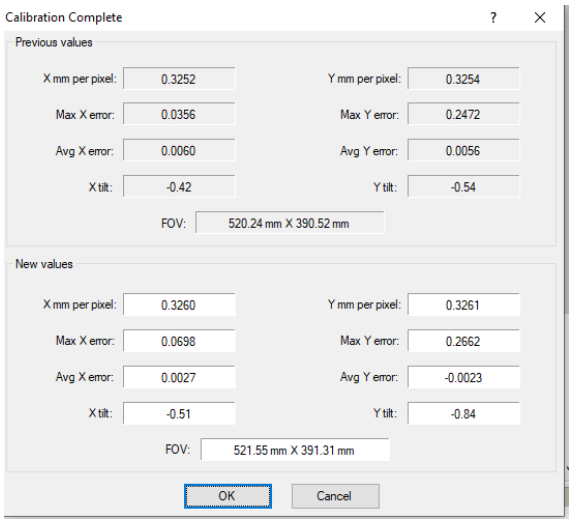

Fig. 5. Summary After Calibration

A. Distortion Correction

Distortion correction enables the camera to detect geometry with higher accuracy, enables the robot to pick with high accuracy, and measures dimensions correctly.

Distortion correction is done similarly in the way as robot vision calibration is done. For image distortion to be done, a grid must be made. The pitches along the horizontal and vertical axis must be equal and the grid must have more than 100 targets in the camera's field of vision. It must cover the entire area of the camera's view. The sheet must be placed at the same height as the working components are placed at.

Fig. 6 shows the distortion sheet used for this project. The camera automatically corrects distortion for the images taken afterwards.

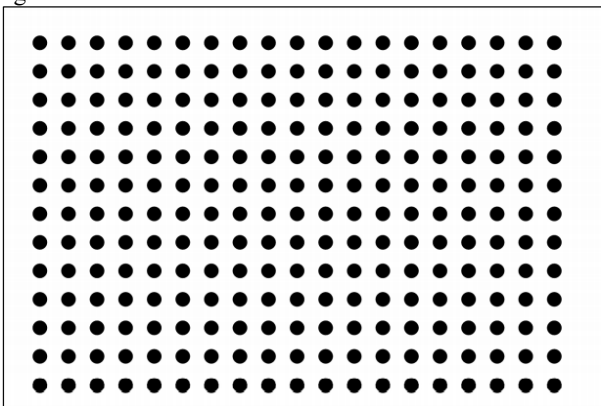

Fig. 6. Distortion Sheet

B. Tool Calibration

Tool Calibration is done in order for the robot to measure how much the tool deviates from a point when it is rotated along a point. This gives the value the robot has to compensate when the tool rotates to pick up a component.

The tool wizard can be opened by opening tools in the top menu and going to robot manager.

In the following step, the tool to teach is chosen, (1 is finger gripper and 2 is magnetic gripper), then next is pressed. The jog window appears. The tool is jogged to a point and the point is taught. Now in the next window rotate the U axis by 90 degrees.

Jog the robot back to the original point with respect to x and y and correct the offset. Afterwards, a summary screen will pop up with the corrected values.

The tool calibration is required to be done only if the gripper hits something, the frame of the robot is shifted or the tool is changed to a new tool different from the existing tools present.

5 Graphical user interface for robot control

The graphical user interface developed in python to control the Robot and PLC without using the included software suite to control the robots. Modbus communication protocol is used as the communication protocol via ethernet except for using TCP/IP to send the name of the component.

A. Pick Window

The pick window of the Graphical User Interface shows the parameters that are used by the robot during the picking process. The following parameters can be adjusted by the operator:

- Component Model
- Offset Correction
- Camera Brightness and Contrast

The load button is used to send the selected string value to the robot. The open software button is used to open the software for teaching a new component. The teach button receives the most recent taught geometry model name from the robot. The sending and receiving of the geometry name uses TCP/IP protocol and socket programming in python to achieve it.

The current row and column display the current position of the robot in the pick tray. It is approximately calculated.

B. Place Window

The place window of the Graphical User Interface shows the parameters that are used by the robot during the place process.

The following parameters can be adjusted by the operator:

- Tray Type
- Offset Correction
- K Value

The load button tells the robot which tray is chosen so the robot can choose the program to accordingly place the component in the slots accurately. The current column and row show

the current position of the robot placing on the tray. The total rows and columns show the total value of the same of the tray.

C. Robot Window

The robot window of the Graphical User Interface. These are the parameters that affect the general operation of the robot. The following parameters can be adjusted by the operator:

- Gripper
- Main Speed
- Single Operation Speed
- Jog Speed
- Jog Step Distance
- Jog Axis
- Free All

The current position displays the current position of the robot. The robot by default runs in auto and disables the jog buttons.TP-Jog is also hidden until auto is pressed to prevent the operator from accidentally triggering jog mode.

Delay hold defines how long the robot should take to pick or place the component from its place on the tray or slot on the place tray.

D. I/O Window

The I/O window of the Graphical User Interface shows the status of robot inputs and the control for outputs. The operator can manually trigger outputs if required for the robot and PLC. The list of outputs is also mentioned for easier remembrance. Conveyor one refers to the pick tray conveyor while conveyor two refers to the placing tray conveyor. Fig. 7 shows the I/O window of the Graphical User Interface.

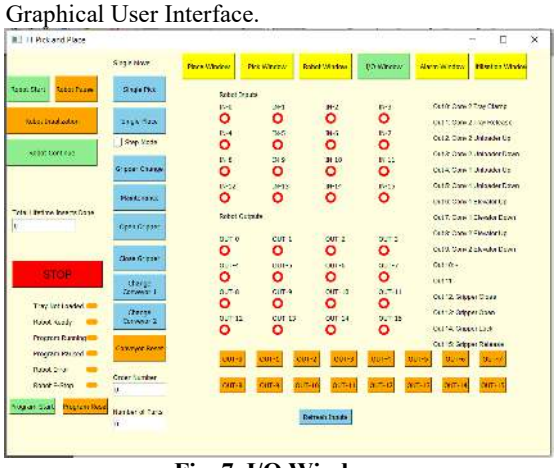

Fig. 7. I/O Window

E. Alarm Window

The alarm window of the Graphical User Interface. This displays any error that occurs in the PLC or robot. The robot stops if a PLC alarm is raised. The text boxes display only the list of current alarms occurring. The reset buttons reset the warnings, clear them from the memory of the robot or plc and re-start the program. The manuals are also given for reference

if needed in the future. The alarm manual shows the list of the possible errors the robot can give which is about 10,000 alarm codes.

This window helps to measure the average cycle time without needing to open the robot software and view the values. This also helps to maintain track of the long-term average performance of the robot and ensure it is consistent for a certain period of time. Fig. 17 shows the alarm window of the Graphical User Interface

6 Testing

The testing phase for a project of this nature is crucial as it reveals the issues that may have been overseen. This also measures the productivity of the arrangement of various other machines crucial to the process such as conveyors and much more. Testing also provides data that is used to optimize productivity and reduce the chances of major downtime while on the production line.

A. Initial Testing Phase

The robot was mounted on a temporary base which was a hindrance for touching the full speed potential of the robot. The maximum safe speed was 40 percent without any serious vibrations or jerks. The maximum rate of change of acceleration and deceleration was only 50 percent, with a maximum possibility of 120 percent for the robot. The picking and placing accuracy were above average.

A cycle time of averaging 3-4 seconds per pick and place was achieved at 40 percent speed by the robot. The camera took 3-5 seconds to identify a maximum of 100 components at a time with an accuracy of about 70-80 percent due to lighting being inadequate. The robot maintained perfect accuracy with an error margin of 0.05mm while picking when the calibration was done properly. Calibration included distortion correction and tool calibration.

B. Calibration

The software of the robot is able to calculate the maximum and average X-axis and y-axis error after calibration is done. As described previously, calibration involves moving the robot arm with a gripper to 9 points within the camera field of vision (FOV).

The first few calibrations had low average errors but a large value of maximum error. The average error being low was not the only factor ensuring good pick accuracy of the component.

Calibration with a high max error of x and y-axis resulted in the gripper not aligning with the centre of the component. This resulted in the gripper slamming onto the pick tray occasionally, failing to pick at the centre of the gripper. As a result, the component tended to get chipped or the gripper pads getting bent. Adjusting the pick offset alleviated the issue of improper calibration. However, this was not a one-fits-all solution as the offset would have to be adjusted for every pick cycle.

The error value was minimized by ensuring the calibration sheet was stuck as flat as possible and the surface was as level as possible. Image operations were done to reduce the contrast of the image and improve the quality of the image improved the calibration accuracy and reduced the error values. Fig. 8 shows the error values that were obtained after several attempts at calibration.

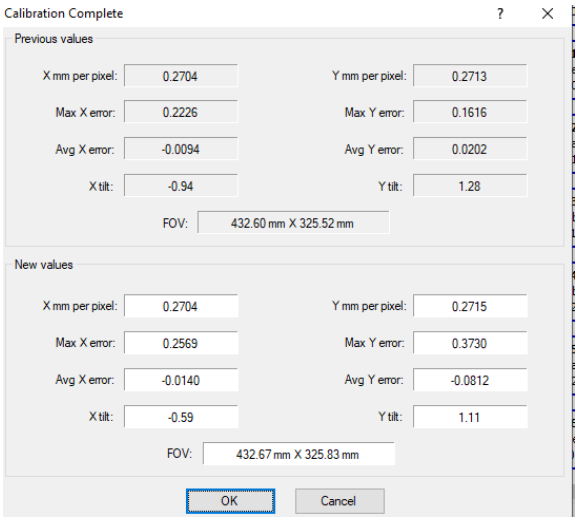

Fig. 8. Ideal Error Values for Calibration

C. Final Testing Phase

The robot was properly mounted onto a base with proper vibration isolators made of rubber and grouting was also done. With the proper mounting of the base, the robot was able to run at full speed with a rate of acceleration and deceleration at a maximum of 120.

The cycle time reduced to 2 seconds per component. The robot was able to maintain the cycle time consistently. The picking and placing accuracy increased as the components were picked and placed onto trays properly held in place. The camera's accuracy was consistently above 85 percent with proper lighting arrangements and maintained the same scan times.

The robot maintained perfect accuracy with an error margin of 0.05mm while picking when the calibration was done properly and with a maximum error value of below 0.3mm. The distortion correction and tool calibration also improved due to proper surface to place the calibration and distortion correction sheets.

D. Teaching Tool Centre Point for Grippers

For the robot arm to pick up a component the robot has to know the offset the gripper has when it rotates at a point. Two types of grippers are used, a two-finger internal diameter gripper and a magnetic gripper. Teaching the tool centre point for the finger gripper can be done easily as the point of contact of the tip of the gripper is a point.

However, for the magnetic gripper, it has a large area of contact with the component which makes it difficult to place the gripper at a single point. This was partially overcome by doing multiple centre point teaching and taking the average of the errors of the teaching done. Also, the outline of the gripper when it was touching a sheet of paper was drawn and after rotation, the robot was jogged till the outline of the previous rotation of the gripper and the current rotation of the gripper met.

Another solution for this was reducing brightness and contrast via the GUI application. The consequence of it is that it takes a few seconds for the brightness and contrast to change due to software limitations.

The final solution for difficult to recognize components was doing image operations, in this case, converting the grayscale image into black and white for easy identification of features

E. Camera Field of View

The camera FOV (field of view) was too large at the height it was mounted initially. This resulted in the camera seeing a much larger than needed and feature identification of the camera features would be less clear.

The height of the camera was then reduced which allowed for a smaller FOV and the ability to see the sharp features of the component more clearly. This aided in improving feature recognition while geometry teaching.

F. High Cycle Times

The target average cycle time for a pick and place operation for the robot was less than 2 seconds. Initial testing at 40 percent of the robot's speed was done, giving a cycle time of 3-4 seconds per component.

After installation onto the production, the cycle time dropped to an average of 2-2.5 seconds per component with the robot running at the full potential of its speed and rate of acceleration and deceleration change.

The final average time achieved was 1.5-1.7 seconds achieved by optimising the rotation of the joints during pick and place operation. Joint 1 and Joint 2 of the robot were optimised for the pick and place cycle. An arc motion for the z-axis was also implemented to reduce the robot needing to wait for Joint 3 to extend to pick or place a component.

The lefty configuration of the robot reduced the movement of the joints while picking and placing components and decreased the cycle time. After a specific point, this went beyond the reach of such a configuration and the robot had to switch to convectional motion for picking and placing components, which was usually near to the ends of the picking tray.

However, due to the high rate of acceleration and deceleration, the place accuracy was compromised, and the reduced parameters increased cycle time to an average of 1.85 seconds for the left configuration and 2 seconds for the convectional movement of all joints.

G. Jerking of Base

The high acceleration and deceleration changes within a very short time span caused major jerking of the base, resulting in movement of the base which would affect the calibration done, the accuracy while picking and placing the components and increase the risk of the robot toppling over. The base is exposed to static and dynamic stress caused by the motion of the robot.

Grouting was done to reduce the jerks and to ensure the base was as much as stable as possible.

H. Sending Negative Values

A negative value cannot be sent directly via Modbus to the PLC due to the use of two's complement. A number range of -32768 to 32767 rather than a range of 0-65535 for a single register of 16 bits is possible for Panasonic Plc. Any number sent beyond 32768 will be considered as in the range -32768 to -1. This was brought to attention when testing the sending of offset values along x, y, z, and an axis for placing the component onto the place tray.

To read negative values from PLC and to send negative values to PLC, any number greater than 32768 must have 32768 subtracted from it to get the negative value required without needing to do two's complement.

I. Sending Strings

Due to the complexity of sending a string via Modbus to the PLC then to the robot, an easier method was to use TCP/IP communication using python socket programming to send a string value directly from Python to the robot.

Socket programming is a method to connect two network nodes to communicate with each other, one socket listens while the other reaches out to the other to initiate a connection. The server becomes the node that listens while the client is the node that reaches out to the server.

Sending data via TCP/IP also required the use of certain characters, being a part of the ASCII protocol, those characters were:

- CR (carriage return)
- LF (line feed).

These are control characters and are usually used to indicate the end of the line. They are used as final characters for each name sent from python to the robot and respectively for receiving data from the robot.

The strings were sent as a byte object, indicated by a b' in front of the string before being sent.

Python has a function to encode and decode byte objects before being sent via TCP/IP.

A memory relay inside the PLC acted as a switch for the robot to open and close the port when data was about to be sent via TCP/IP protocol.

J. Components Flying Off Gripper

Components tended to fly off the magnetic and finger gripper while running at maximum possible speed and acceleration and deceleration times. This was caused by the gripper having a tapered profile. The tapered profile had a lower surface contact area, resulting in a lower grip on the component surface. Due to the lower surface of contact, components that had a tapered inner hole end up flying off the gripper when the gripper slows very fast when the robot is about to place. The component which flew off tended to damage other components causing surface chipping and micro cracks.

The gripper has to be redesigned with a new profile in order to increase the area of contact while picking up the component in order to reduce the chances of the component flying off when a rapid change of speed of the gripper occurs.

K. Components Flying Off Gripper

The robot is able to pick and place an average of 1850 components per hour with an average cycle time of 1.85 seconds per component. The average components per hour include the time the conveyors take to move the trays and the time taken by the camera to scan the pick tray before picking. The camera can only scan a maximum of 100 components per scan.

7 Conclusion

Automation of a pick and place process done by integrating SCARA robot in the production line in order to increase productivity and to reduce the waste of manpower for

monotonous tasks. The camera, robot, vision controller and a PLC have been successfully linked up with a Python application successfully. The robot has been successfully installed onto the production line. Extensive testing has been done to iron out the minor issues that were missed in testing before the robot was installed. An initial cycle time of 3.5 seconds was reduced to 2.5 seconds after installation onto the production line. Further optimisation gave a final cycle time of averaging 1.85 seconds per component to pick and place. The Python Graphical User Interface has been tested in the production line for crashes and to ensure the application does not crash in crucial situations. The Graphical User Interface application has successfully eliminated the need to use the robot software and PLC software to control the working of the robot and PLC.

References

[1] Mohd. Nayab Zafar, J. C. Mohanta, Alok Sanyal, "Design and Implementation of an Autonomous Robot Manipulator for Pick & Place Planning", IOP Conf. Series: Materials Science and Engineering, 2019

[2] S. Suri, A. Jain, N. Verma and N. Prasertpoj, et al "SCARA Industrial Automation Robot," International Conference on Power Energy, Environment and Intelligent Control (PEEIC), pp. 173-177, 2018.

[3] Khin Moe Myint, zawmin Min Htun, Hla Myo Tun, "Position Control Method for Pick and Place Robot Arm for Object Sorting System", International Journal of Scientific & Technology Research Volume 5, Issue 06, June 2016.

[4] Mark Summerfield, "Rapid GUI Programming with Python and Qt: The Definitive Guide to pyqt Programming", Pearson Education, 2007.

[5] Joshua M. Willman, "Beginning PyQt: A Hands-on-Approach to GUI programming", Apress, 2020.

[6] Yuan-Che Hsu, Tu-Cheng (Tw); Du-Xue Zhang, Shui-Ping Wei, "Robot Calibration System and Calibrating Method Thereof", Patent No- Us 2011/0320039 A1. Patent Date - Dec. 29, 2011.

[7] Masatoshi Ono Christoph Meyerhoff, "Horizontal Articulated Robot", Patent No - US 2010/0050806 A1, Patent date- Mar. 4, 2010

[8] Seiko Epson Corporation, "EPSON RC+ 7.0 (Ver.7.5) SPEL+ Language Reference Manal", Rev.1, 2020.

[9] Seiko Epson Corporation, "SCARA ROBOT T series Manipulator Manual", Rev.12, 2020.

[10] Seiko Epson Corporation, "Epson RC+ 7.0 Option Vision Guide 7.0 (Ver.7.3) Hardware & Setup", Rev.3, 2020.