DESIGNING AND CREATING A PALLETIZING ROBOT COMBINED WITH IMAGE PROCESSING

Van-Tan Nguyen[1], Dinh-Thai Hoang [2*], Vu-Thanh-Phuong Le[3], Minh-Giang Tran[3], Truong-Phu Nguyen[3], Ngoc-Hung Hoang[3], Van-Dai Le[3], Huy-Vu Tran[3]

[1] Honda Plus Vietnam Co., LTD
307/1 – 7A Road – Amata Industrial Park – Long Binh Ward – Bien Hoa City – Dong Nai Province, Vietnam
[2] Azbil Vietnam Co., LTD.
Phung Khac Khoan Street, No. 28, Da Kao ward, District 1, Ho Chi Minh City, Vietnam
[3] Ho Chi Minh city University of Technology and Education (HCMUTE)
Vo Van Ngan, 01, Ho Chi Minh city, Vietnam
***** Corresponding author. E-mail: h.thai.2i@vn.azbil.com

Abstract: In this paper, we propose a solution of designing and building a three-order 4D magician robot arm to palletize goods. Inverse kinematic of system is calculated. Thence, we design a suitable robot arm which can utilizing camera to palletizing goods. PID controller is used to move operating point. Simulation and experimental results are shown to prove the well operation of this robot arm.

Keywords: robot arm; PID control, inverse kinematic; palletizing; magician 3D.

1. Introduction

Scientific and practical significance of topic of robot arm is importance in today's rapidly advancing technological landscape. Robot arms **[1]**, also known as robotic manipulators, are mechanical devices designed to mimic and replace human arm movements and capabilities. These robotic arms play a vital role in various fields, including manufacturing, healthcare, space exploration, and more.

Fig. 1. Car assembly robot [3]

Moreover, integration of machine learning and artificial intelligence with robot arms opens up new possibilities. By enabling robots to learn from data and adapt to different scenarios, they can become more versatile and capable of handling complex tasks autonomously. This has implications not only in industrial settings but also in areas such as search and rescue operations, disaster response, and environmental monitoring. Seeing this great potential, robots are the best solution in many countries to boost the country's economy.

Control the robot to pick and arrange objects to pallet based on informations extracted from camera. In addition, an interface for operator can monitor and intervene in robot's operation process.

Fig. 2. Objective of the topic

This project focuses on the research and development of Robot Magician 4 DoFs system [4]. The primary objective is to automate the arrangement of goods from conveyor belts to pallets, utilizing position data provided by the camera [5].

2. Theoretical Basis

2.1. Operating Explanation

When starting, robot is returned to home position by detecting limit switch. Object is placed on the conveyor belt, and it is taken to location where coordinates are sent from Raspberry via Arduino. Arduino will send pulses to stepper motor to control robot moving to object position. After the object is picked up, robot returns to home position again and then moves to pallet positon, rotates object and places it on pallet surface. Robot will repeat this work until there is no more object inside sending coordinate area or there is a stop command from operator or pallet is full.

2.2. Kinematics

The first thing to do when solving robot kinematics problem is to set appropriate coordinate system for robot configuration. This makes calculating forward and inverse kinematics accurate and easy. From coordinate system above, we get DH-Table (Tab. 1) for 3 main joints:

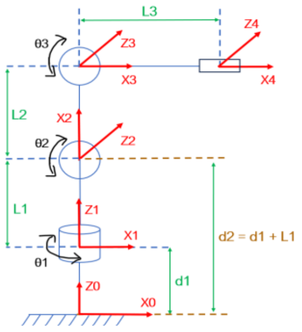


Fig. 3. Coordinate system of Magician robot

Tab. 1. Denavit–Hartenberg parameters table of Magician robot

DOF	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	d2	θ1
2	0	-90°	0	-θ2
3	L2	0	0	θ3
4	L3	0	0	0

where: a_i : the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measure along $\hat{X}_i$; α_i: the angle from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measure about $\hat{X}_i$; d_i : the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measure along $\hat{Z}_i$; θ_i : the angle from $\hat{X}_{i-1}$ to $\hat{X}_i$ measure about $\hat{Z}_i$

Forward kinematics

Calculate forward kinematics, helping to precisely control the desired position by changing the angle of each robot joint.

The general formula:

$$ {}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

From the established DH table, we have the results of the transposition matrix ${}^0_1T$, ${}^1_2T$, ${}^2_3T$, ${}^3_4T$

The result of ${}^0_4T$ has form:

$$ {}^0_4T = \begin{bmatrix} x11 & x12 & x13 & Px \\ x21 & x22 & x23 & Py \\ x31 & x32 & x33 & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The position of End-effector:

$$ {}^{0}_{EE}P = \begin{bmatrix} P_X & P_Y & P_Z & 1 \end{bmatrix}^T \quad (3)$$

where: $P_X = c\theta_1\left(L_2 c\theta_2 + L_3 c\left(\theta_2 - \theta_3\right)\right)$;

$P_Y = s\theta_1\left(L_2 c\theta_2 + L_3 c\left(\theta_2 - \theta_3\right)\right)$;

$P_Z = d_2 + L_2 s\theta_2 + L_3 s\left(\theta_2 - \theta_3\right)$

Inverse kinematics

Inverse kinematics is a problem employed to ascertain the values of joint angles when the position of the endpoint is known. Our group utilizes geometric methods to determine the values of the theta angles in this context.

Finding θ_1:

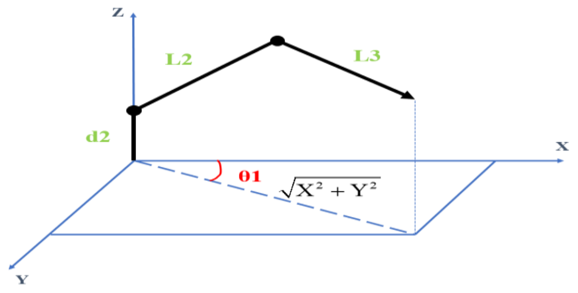


Fig. 4. Describe θ_1 geometrically

Thus, the value of θ_1 is equal:

$$\theta_1 = \tan^{-1}(Y/X) \quad (4)$$

Finding θ_2:

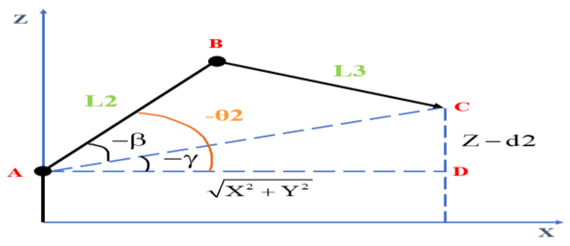


Fig. 5. Describe θ_2 geometrically

Then, θ_2 is equal to

$$\theta_2 = \beta - \gamma \quad (5)$$

where $\gamma = \tan^{-1}\left(\frac{Z-d_2}{\sqrt{X^2+Y^2}}\right)$;

$$L_3^2 = AC^2 + L_2^2 + 2L_2 AC\cos\beta;$$

$$\cos\beta = \frac{X^2+Y^2+\left(Z-d_2\right)^2+L_2^2-L_3^2}{2L_2\sqrt{X^2+Y^2+\left(Z-d_2\right)^2}}$$

Finding θ_3:

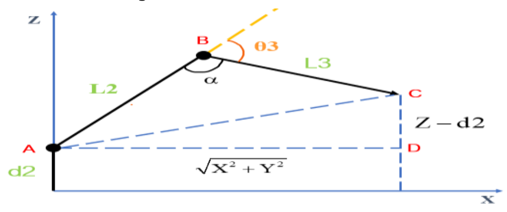


Fig. 6. Describe θ_3 geometrically

$$\theta_3 = \tan^{-1}\left(\sin\theta_2 / \cos\theta_3\right) \quad (6)$$

where: $AC = \sqrt{X^2+Y^2+\left(Z-d_2\right)^2}$;

$$\cos\theta_3 = \frac{X^2+Y^2+\left(Z-d_2\right)^2-L_2^2-L_3^2}{2L_2L_3}$$

Tool position

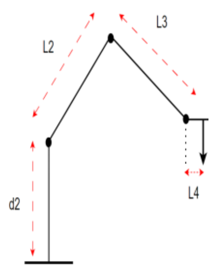


Fig. 7. Robot dimension

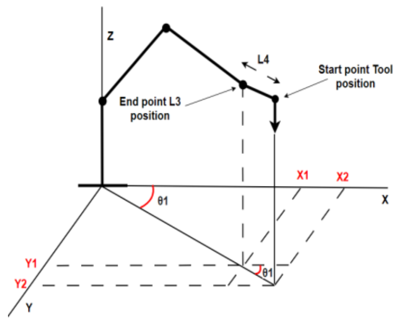


Fig. 8. Tool position (1)

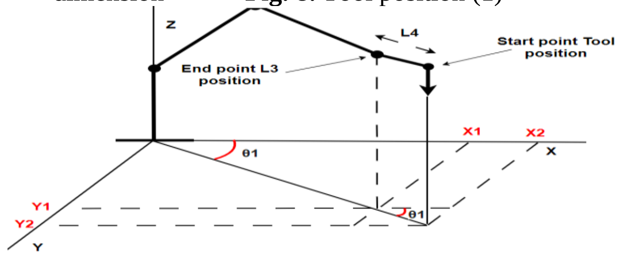


Fig. 9. Tool position (2)

We see that:

$$\begin{cases} \Delta X = L_4 \cos\theta_1 \\ \Delta Y = L_4 \sin\theta_1 \end{cases} \tag{7}$$

Then, we obtain

$$\begin{cases} X_1 = X_2 - \Delta X \\ Y_1 = Y_2 - \Delta Y \end{cases} \tag{8}$$

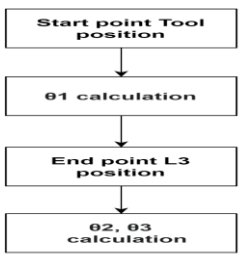


Fig. 10. Position processing

With X1, Y1 and Inverse Kinematic, we can find exactly θ2, θ3 for robot. Because, (x_2,y_2) coordinate and (x_1, y_1) coordinate lie on a straight line. Therefore, θ_1 is calculated from (x_2, y_2) or (x_1, y_1) are same.

3. Hardware

3.1. Image Processing

Camera distortion refers to inherent imperfections in capturing images due to camera lens's optical characteristics. These distortions often manifest as radial or tangential distortions, causing objects in image to appear warped or misshapen. Radial distortion results in a stretching or squeezing effect towards or away from center of the image, while tangential distortion occurs as a shift in image's perspective. Presence of distortion can significantly impact accuracy of measurements and reliability of object recognition in computer vision applications. Addressing these distortions is crucial before applying camera calibration to ensure precise and reliable image processing and analysis in various applications, such as object recognition and robotic automation.

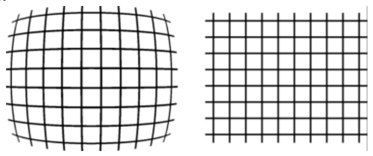


Fig. 11. Distortion in camera with straight lines

We use a checkerboard patterns for camera calibration, corners of squares on checkerboard are ideal for localizing them because they have sharp gradients in two directions. In addition, this method is widely used worldwide, so the group is easily accessible.

Fig. 12. Camera calibration with checkerboard

By keeping camera constant and photograph checkerboard pattern at different orientations. We have got camera distortion matrix:

$$\begin{bmatrix} 1.201912191422151333e{+}00 \\ -1.981891000020094040e{+}01 \\ 1.550021049206086465e{-}02 \\ -1.608339622550517337e{-}02 \\ 1.010980712754502804e{+}02 \end{bmatrix}^T \tag{9}$$

Thus, information about camera position and orientation, aiding in accurate prediction of an object movement in scene are provided in 2 matrices above.

After extracting the camera's features, we need to determine the best location to place camera.

Results after Applying the Algorithm:

In image below, landmarks are positioned at fixed locations on conveyor belt to serve as reference points for measurement boundaries. Although these positions have been measured and set to be uniform and parallel, during camera detection, deviation among these points reaches up to 5-6 pixels.

Coordinates before applying undistortion matrix are listed as follows: A (125,38); B (124,358); C (1098,359); D (1103,38).

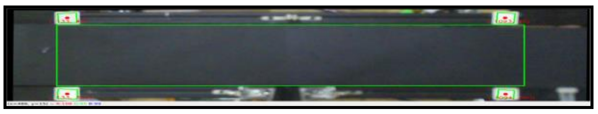


Fig. 13. Coordinate of reference points after undistorted

Computer vision program using OpenCV and Tkinter (a GUI toolkit for Python) to capture video from a webcam, process the frames, and display the results in a GUI window.

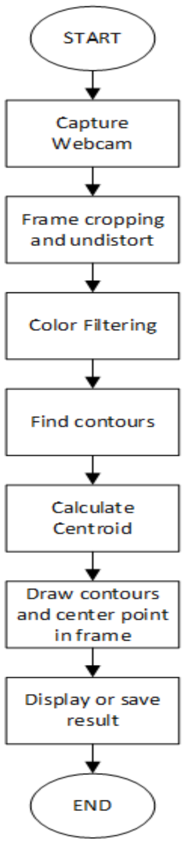


Fig. 14. Image processing steps

Default camera settings capture an extended range in image, leading to unnecessary processing of excess data and extraction of false information. To address this, it is essential to crop frame to a specific region of interest. This proves beneficial when our focus is solely on analyzing or processing a specific area within overall scene.

Process initiates when an object is positioned on conveyor belt. Subsequently, frames are consistently captured from camera. In subsequent step, preprocessing operations are executed on frames to ready data for object detection by transforming frame into HSV color space.

Following conversion to HSV color space, we employs morphological operations to execute erosion on input image. Subsequently, an opening image method is applied (dilation on result of erosion). These transformations serve to diminish unwanted noisy elements, resulting in creation of intended grayscale image.

Subsequently, we proceeds to identify contours within designated color areas, where each contour signifies a product that requires classification within frame. In this stage, we utilizes findContours function to identify contours of objects.

After identifying and drawing bounding boxes around objects, to enhance accuracy and eliminate misidentification in brighter lighting conditions or strange objects enter frame, we have implemented conditions for locating center of each object. Object's area is calculated using cv2.contourArea(), and number of edges in approximated contours via cv2.approxPolyDP() must be 4, ensuring object is square and matches the reference object's area. In summary, two conditions must be met before finding center of detected object:

+ cv2.contourArea() >= Tested Area for object

+ cv2.approxPolyDP() == 4 (to make sure its square shape)

When an object meets both conditions, indicating it matches reference object, center of object is determined using Image Moment method. Function cv2.moment() will be applied directly on contour of detected object.

By summing up the product of the x-coordinate and y-coordinate of each pixel and its intensity value. Centroid of an object in image can be calculated using formulas:

$$Centroid_x = \frac{m10}{m00} \quad Centroid_y = \frac{m01}{m00} \tag{10}$$

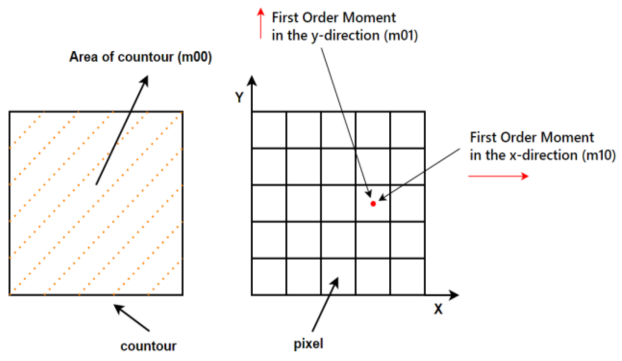

First Order Moment in the y-direction (m01)
Area of countour (m00)
Y
First Order Moment in the x-direction (m10)
X
countour
pixel

Fig. 15. Image moment method

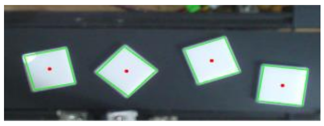


Fig. 16. The object is determined centroid

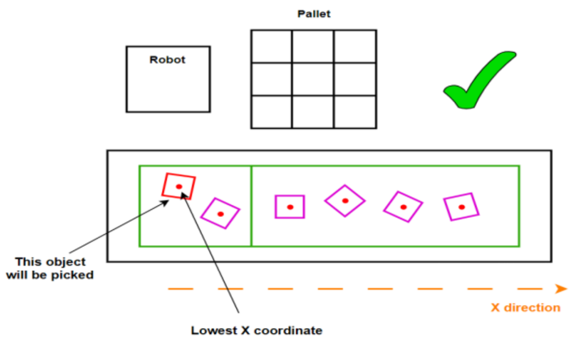


Fig. 17. Object order sort solution

Multi – Color Detection:

To enhance capabilities of robot, we has developed an additional algorithm for detecting and classifying colors of objects to be grasped. To implement this feature, it is necessary to expand HSV color range. Initially designed for white color, through gathering available information and practical experimentation, we have obtained extended range of limit parameters for white color in HSV color space as [0:180,0:180,160:255].

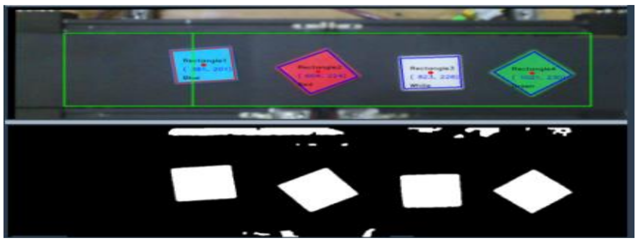


Fig. 18. Result of multi-color detection

To expand color detection range, we added upper and lower thresholds for HSV color range for green, blue, and red hues. Obtained results are as shown in illustration. Subsequently, we proceed to identify accurate color values to be sent to Arduino and map them to designated positions for each color.

To identify color again in frame after being detected, a region of interest (ROI) is identified and extracted from HSV (Hue, Saturation, Value) representation of image. which captures HSV values within a specified rectangle defined by top-left corner coordinates (x, y) and dimensions w (width) and h (height). Subsequently, average HSV values for this region are computed using the NumPy function np.mean(). Mean is calculated independently for each channel (Hue, Saturation, Value) across entire region. Finally, resulting average Hue (h), Saturation (s), and Value (v) are stored for further analysis. This process enables a focused examination of specific image regions in terms of their color information, contributing to a more detailed understanding of the visual content.

Deviation Angle Calculation

Finally, we determines deviation angle of object concerning its placement on pallet. cv2.convexHull() function is employed to facilitate identification of deflection angle. In computational geometry, convex hull of a point set is smallest convex set encompassing all points. For instance, when a square object is captured by camera, convex hull function connects four edge points, arranging them according to Graham scan rule. Hull [0,1,2,3] named as A, B, C, D.

Algorithm begins by identifying point with lowest y-coordinate. If there are multiple points with same lowest y-coordinate, point with lowest x-coordinate among candidates is selected. Therefore, we establish:

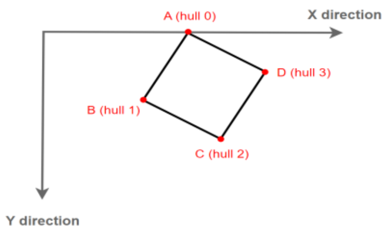


Fig. 19. Convex hull algorithm

Group takes hull [1] - hull [2] to create a vector $\overrightarrow{BC}$, this vector will be matched an angle with the x axis. This angle is angle of deviation between the object and its position on the pallet.

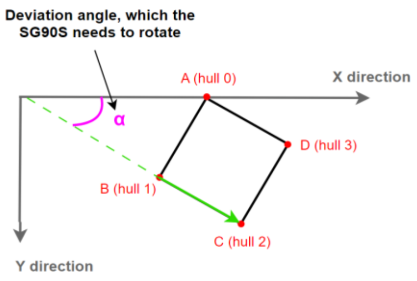


Fig. 20. The angle of deviation

After obtaining deviation angle of object relative to X-axis, it is necessary to calculate deviation angle after moving from conveyor belt position to pallet. When object is gripped by robot, object's angle relative to robot arm remains constant and is augmented by rotation angle of theta1 from gripping position to release position. Therefore, deviation angle of object relative to horizontal axis (X-axis) is equal to measured deviation angle at conveyor belt position minus rotation angle theta1.

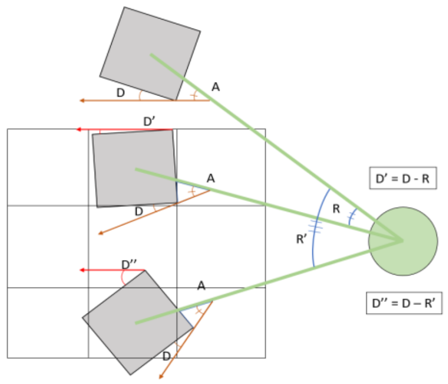


Fig. 21. Describes angle of deviation of object picked-up

Another issue with the 4-DOF robot arm is that the servo motor can only rotate in one direction and is limited to a range of 0 to 180 degrees. Therefore, initial position of servo motor is set at 90 degrees and adjusted by adding or subtracting deviation angle of object from pallet after being calculated from conveyor down.

Formula for calculating rotation angle of object:

$$D' = D - R \tag{11}$$

- D: Deviation angle of the object on the conveyor belt
- R: Rotation angle of theta1, bringing the object from the conveyor to the placement position
- D': Deviation angle of the object after being moved

When transfer rotation angle to control SG90s servo motor, deviation angle D of object always falls within range of [0-90] degrees, and delta rotation angle R of robot lies within range approximate of [60-120] degrees. Therefore, deviation angle D' also falls within range of [-120 30] degrees. In cases where absolute value of D' exceeds 90 degrees, we continue by subtracting absolute value of D' from 90 degrees to obtain a smaller deviation angle of adjacent side to X-axis.

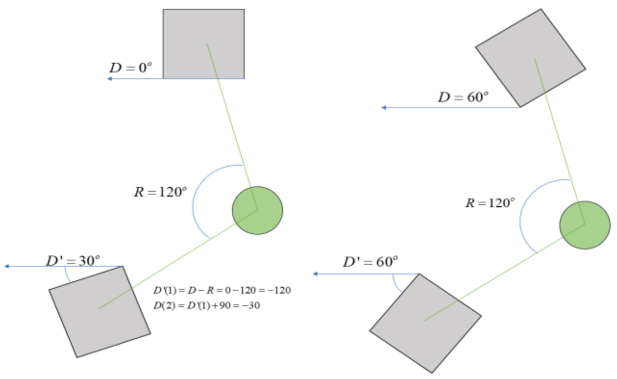


Fig. 22. Demonstration of deviation angle calculation (1)

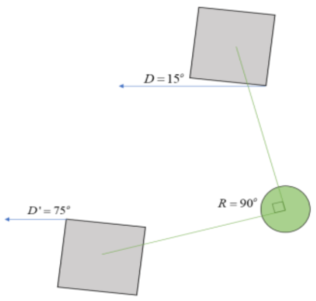


Fig. 23. Demonstration of deviation calculation (2)

Algorithm Verification:

Verifying accuracy of algorithm adjusting deviation angle of object, we conducted tests by placing objects with randomly assigned deviations on conveyor belt and positioning them at three different locations on pallet. Positions on conveyor belt created different Delta Theta 1 angles to test algorithm in various scenarios as listed above.

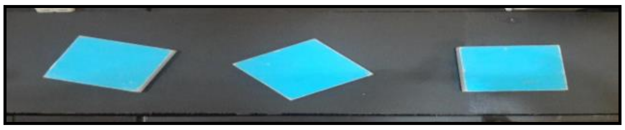


Fig. 24. Test objects for deviation angle correction

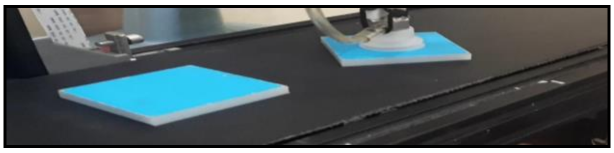


Fig. 25. Robot started picking object before rotate 4th degree

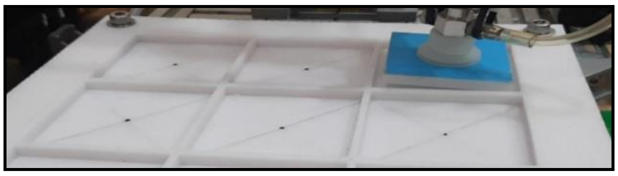


Fig. 26. First object be rotated and putted on pallet

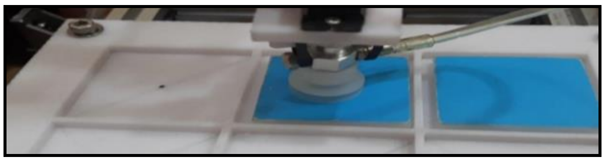


Fig. 27. Second object be rotated and putted on pallet

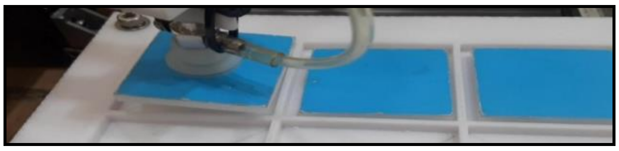


Fig. 28. Third object be rotated and putted on pallet

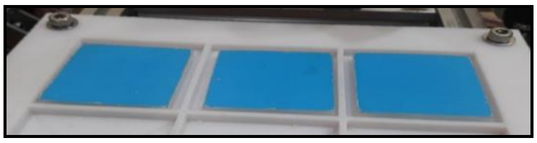
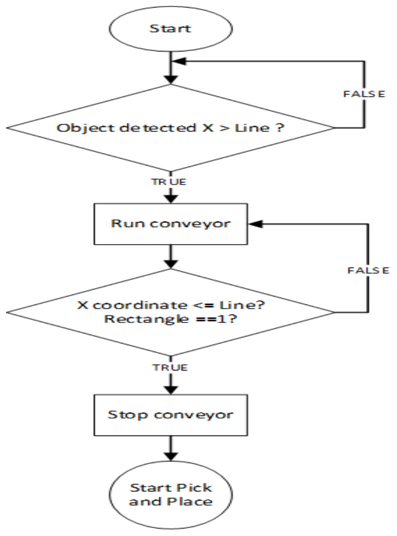
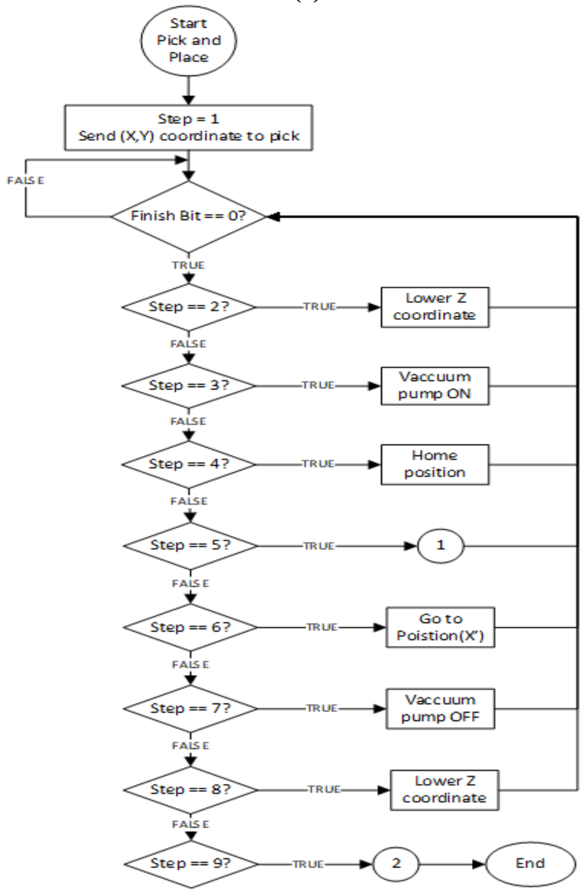


Fig. 29. Total result of objects be rotated and putted on pallet

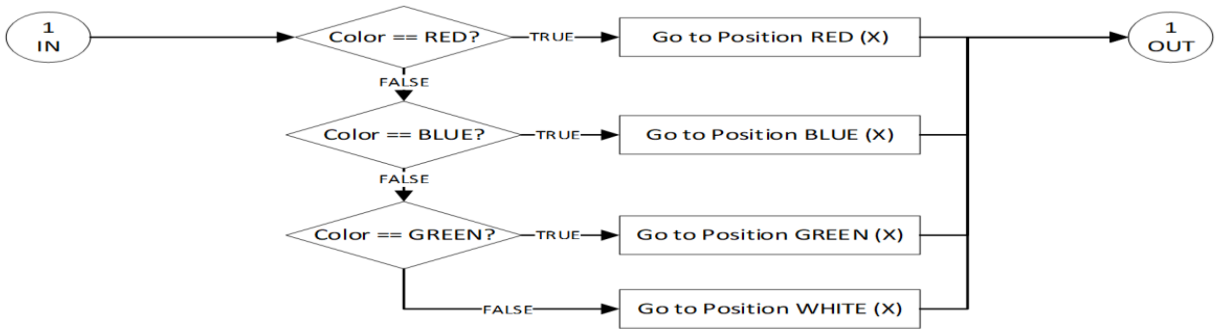
When having center of object, next problem is synchronizing the coordinates of the object on the conveyor belt with the robot. To do this, we manually measured from the center of the robot to median line of conveyor belt. To convert coordinates from frame to real-world coordinates accurately in relation to robot arm, we calculated real-world coordinates of objects on conveyor belt by measuring correlations in image, identifying landmarks on conveyor belt.



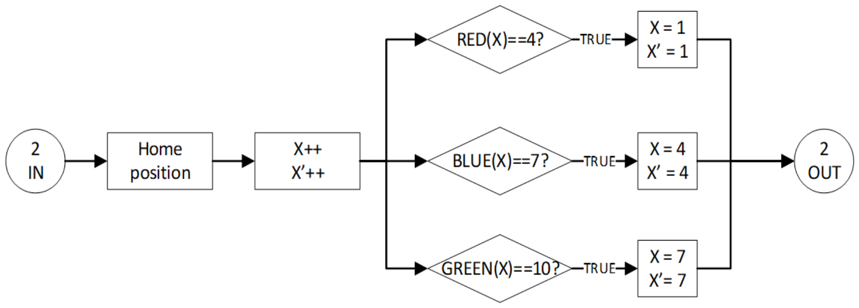
(a)



(b)



(c)



(d)

Fig. 30. Operating diagram of robot arm

3.2. Mechatronics Structure

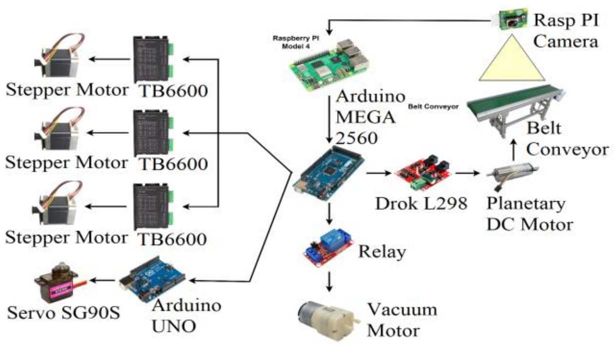


Fig. 31. Hardware structure

4. Simulation

We draw a robot with basic mechanical structures, and these links must have the same length as the real hardware. Then, converting this virtual robot into blocks in Matlab Simulink to prepare for simulating robot.

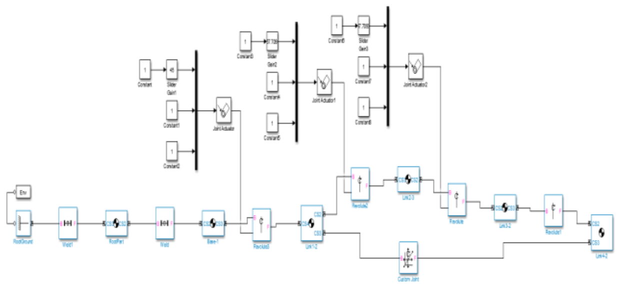
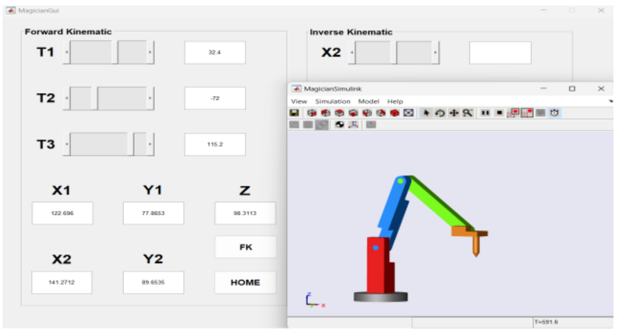


Fig. 32. Simulink model

When Simulink model is completed, we build a Matlab guide to deploy the kinematic of robot on screen and observe position change of robot while changing value of theta angles.



Forward Kinematic

T1 32.4

T2 -72

T3 115.2

X1 122.696

Y1 77.8653

Z 98.3113

X2 141.2712

Y2 89.6536

Inverse Kinematic

X2

T=591.6

Fig. 33. Simulating robot

Comment: After simulating on matlab, we see that,

- Positive direction **(+)**: Moving from left to right
- Negative direction (-): Moving from right to left

The hardware dimensions are:

d2 = 116; L2 = 140; L3 = 140 (12)

Simulation results are

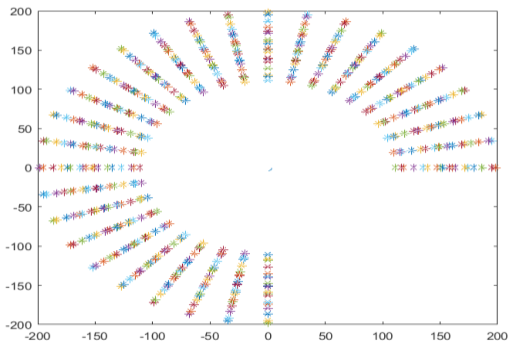


Fig. 34. X-Y view working space

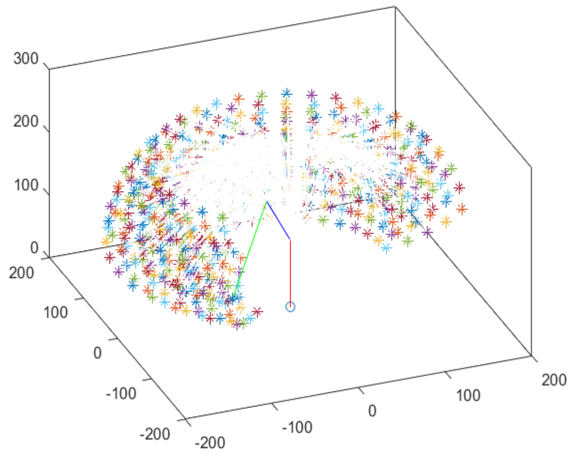


Fig. 35. X-Y-Z view working space

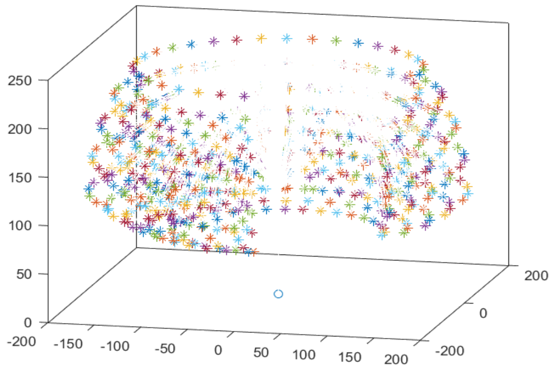


Fig. 36. Points where robot can reach

These are symbolic points to describe Robot workspace, because we take high step to draw faster in Matlab software, lead to have a large space between points.

Comment: Drawing operating space can help visualizing installation location for entire system.

Robot Trajectory Planning

Based on theory, we chose a 3rd order function to find orbital for robot. And problem is to draw a straight line from the orbit starting position (robot is in home form but oriented at position 1 on pallet) to positions on pallet. Here, we took a position on pallet to conduct simulations and trajectory planning calculations.

- Orbit starting position:

$$\begin{cases} x_{tool} = 160.0947(mm) \\ y_{tool} = -35.5766(mm) \\ z_{tool} = 267(mm) \end{cases} \Rightarrow \begin{cases} x_{1,3} = 136.6662(mm) \\ y_{1,3} = -30.3702(mm) \\ z_{1,3} = 267(mm) \end{cases} \quad (13)$$

- Orbit end position:

$$\begin{cases} x_{tool} = 225(mm) \\ y_{tool} = -50(mm) \\ z_{tool} = 100(mm) \end{cases} \Rightarrow \begin{cases} x_{1,3} = 201.5715(mm) \\ y_{1,3} = -44.7936(mm) \\ z_{1,3} = 100(mm) \end{cases} \quad (14)$$

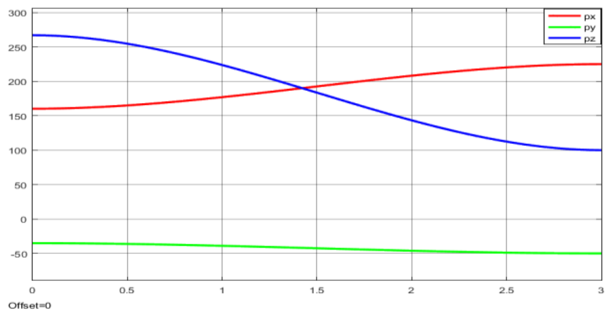


Fig. 37. X, Y, Z position

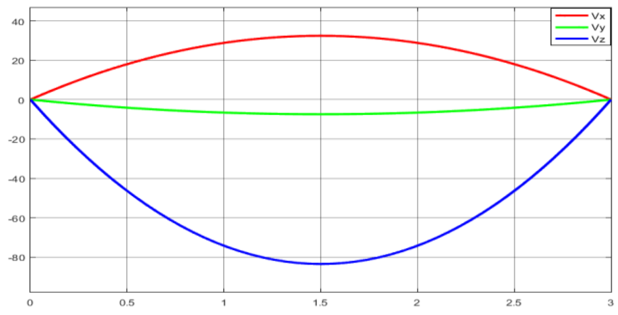


Fig. 38. X, Y, Z velocity

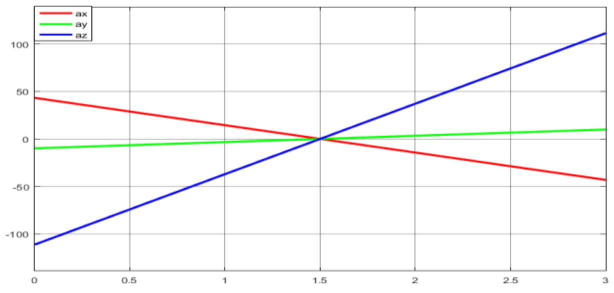


Fig. 39. X, Y, Z acceleration

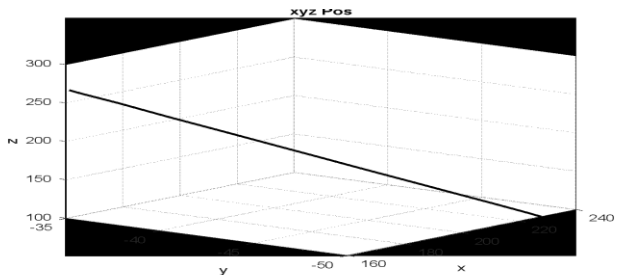


Fig. 40. Straight line from theoretical calculation

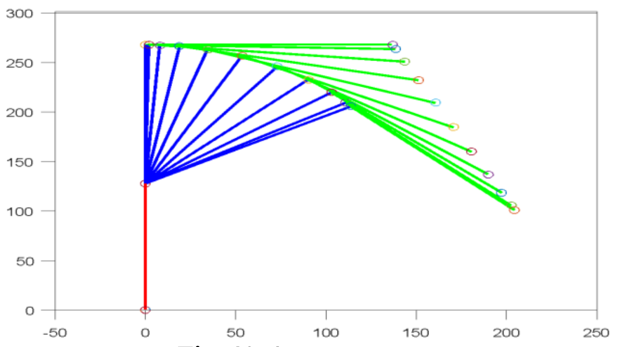


Fig. 41. Arm movement

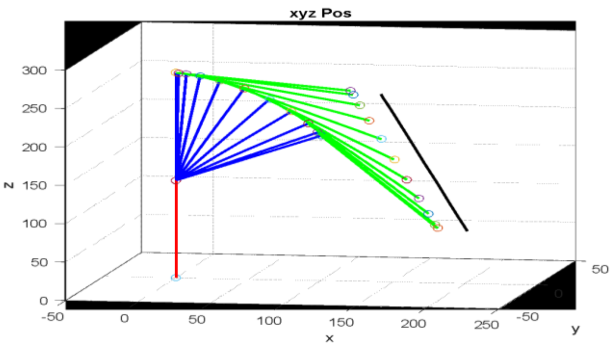


Fig. 42. Orbit in space

Verify the Response of the PID Controller

We built a PID controller for the conveyor motor on a microprocessor, taking advantage of Arduino monitor, we drew a chart to verify controller's response.

Objectives set for PID controller:
- Fast output response, under 2 seconds.
- Fast settling time, under 3 seconds.
- No overshooting beyond 5% when there is an abrupt change in velocity.
- Settling time 1.8 seconds.

With the set requirements, we tried and errored many parameters, and select PID parameters as follows: Kp = 20, Ki = 15, Kd = 0.001. Controller results are shown in the following graphs.

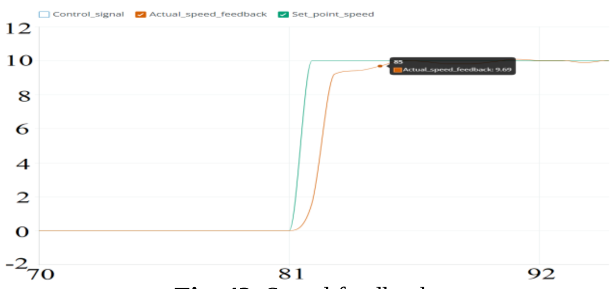


Fig. 43. Speed feedback

Comments:

-Rise time: (from sample point 11 to 12 – sampling time: 0.6s) 0.6 seconds.

-Overshoot: Overshoot is 1%, where the overshoot is 10.09, and the setpoint value is 10.

While the conveyor belt is operating, the speed will change continuously to avoid wasting time for the robot to wait for the object to arrive. Thus, the controller's response must keep up with the change.

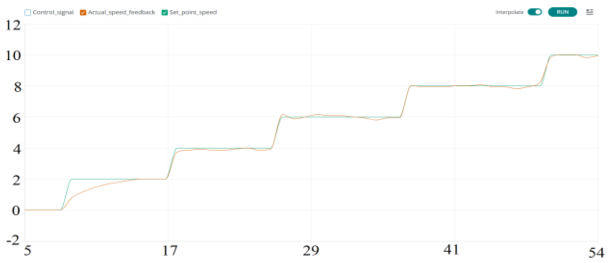


Fig. 44. Setpoint and feedback went up gradually from 0-2-4-6-8-10 reps/min

After multiple experiments, PID set has somewhat met the specified conditions, aiming to limit overshooting when the speed changes abruptly, as well as providing a quick response time to the system's control. However, in cases of significant changes, overshooting still exists, although not high, it will still affect the operational quality. It requires additional tuning time with the PID parameters or combined control methods.

5. Experiment

Fig. 45. Robot Magician Model in this project

Demonstration of Actual Operation during Real-Time Execution

During operation, camera continually detects objects on conveyor belt and brings them to specified position (orange marker) to send information about angle deviation and coordinates of object to be grasped:

Robot arm will move to coordinates of object, lower its height, and initiate the vacuum motor:

Through the measurement process, we obtained an error table for gripping coordinates after 10 trials, as follows:

Tab. 2. The table records the experimental results of the deviation position

Task	1	2	3	4	5	6	7	8	9	10
Deviation X axes (mm)	2	3	1	2	3	3	1	3	2	1
Deviation Y axes (mm)	2	2	2	1	1	0	2	2	0	2

Checking the Order Distinction and Continuous Operation:

After completing the pick-and-place process for one object, the next step involves testing the camera's ability to differentiate the order by trying multiple objects simultaneously on the conveyor belt. As shown in the images, the robot successfully selected the objects in the correct order and continued to move the objects on the conveyor belt to the waiting queue for the next pick.

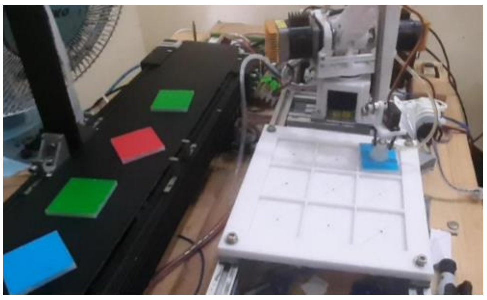


Fig. 46. Robot picking object base on color

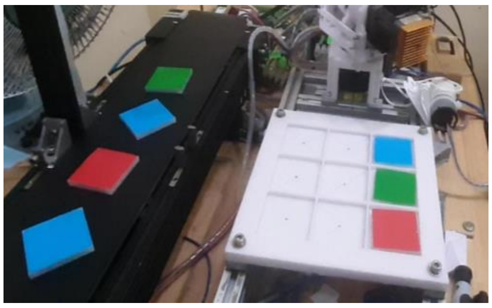


Fig. 47. Robot picking object base on color (1)

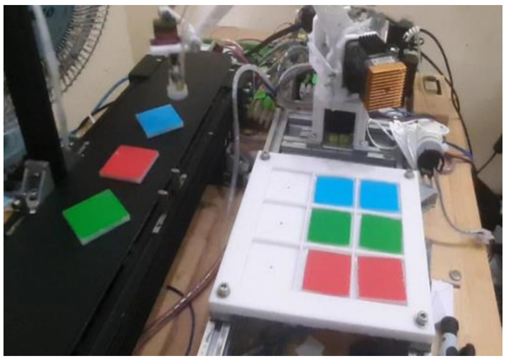


Fig. 48. Robot picking object base on color (2)

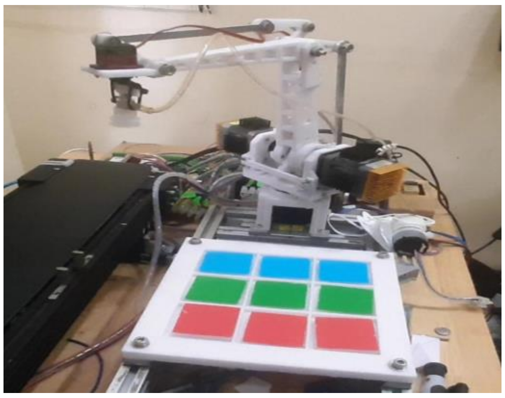


Fig. 49. Robot finish picking object base on color

Following experiments, system seems to work quite steadily, but there are still some classification errors. These errors stem from inaccuracies in mechanical design and position calculations, causing deviations during operation. Although system is generally stable, there's room for improvement in both mechanical structure and position calculations to address these inaccuracies.

6. Conclusions

In study, we has successfully completed the project with notable results: dynamic calculations, 3D-printed hardware design, and creation of a unified monitoring and control interface for both robots on a single platform have been accomplished. The system operates stably, can run continuously, and performs automated tasks based on pre-programmed processes. The interface, crafted using the Tkinter library for real-time monitoring and adjustment of parameters and operational modes, has significantly increased both flexibility and accuracy. With achievements obtained, this sudy provides a practical system model that can be applied in automobile palleting missions in plants, industrial facilities. However, we still faces some shortcomings in applying PID control and executing trajectory planning on robot for smoother operation. Additionally, there is a need for optimization in both hardware and software design to improve communication between devices, thereby enhancing the operational speed of both the robot and system.

7. Acknowledgement

We want to give thanks to PhD. Xuan-Ba Dang (Faculty of Electrical and Electronics Engineering – HCMUTE) due to his supervision. We also want to give thanks to PhD. Van-Dong-Hai Nguyen for his help in editing this work.

8. References

[1] Chaudhari A. et al: "Development of Robotic Arm Prototype", 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), India, 2023, pp. 1053-1057, 2023.
[2] Kim G. et al: "Case studies of a industrial dual-arm robot application", 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Korea (South), pp. 301-302, 2017.
[3] Yamada Y. et al: "Development of multi-arm robots for automobile assembly", Proceedings of 1995 IEEE International Conference on Robotics and Automation, Japan,, pp. 2224-2229 vol.3, 1995.
[4] Ribagin S. et al: "Generalized net model of the DOBOT Magician robot functionalities", IEEE 11th International Conference on Intelligent Systems (IS), Warsaw, Poland, 2022, pp. 1-4, 2022.
[5] Villaverde L. et al: "Camera Calibration Algorithm for Industrial Robot", 2022 IEEE 8th Information Technology International Seminar (ITIS), Indonesia, pp. 41-44, 2022.