The Graphical User Interface for Controlling Delta Robot Movement through G-Code

Riska Analia [1], Nur Afif Gozali [2], Daniel Sipahutar[3], Susanto[4], Eko Rudiawan Jamzuri[5]

{ riskaanalia@polibatam.ac.id [1], afifghoz23@gmail.com [2], danielsipahutar19@gmail.com [3], susanto@polibatam.ac.id [4], ekorudiawan@polibatam.ac.id [5] }

Department of Electrical Engineering, Politeknik Negeri Batam, Batam, Indonesia

Abstract. This research aims to develop a Graphical User Interface (GUI) to control the delta robot using the G-Code command in real time. The proposed mechanical design of the delta robot adopts a parallel arm mechanism. As the primary controller, the Arduino Uno has been chosen as a bridge to translate the joint command from the computer to the robot stepper motor. Furthermore, we proposed the G-Code command to control the delta robot end effector. The system interface was carried out using the C# programming language and the.NET frameworks. As G-Code translation, we proposed an inverse kinematics equation derived from a trigonometric to decode command into joint movement. Finally, the experiment was carried out in real-time to verify our interface. As an experimental result, the proposed system successfully translates the G-Code command into end effector movement.

Keywords: delta robot, parallel arm, G-Code, inverse kinematics, graphical user interface.

1 Introduction

Recently, the delta robot has become one type of pick-and-place robot commonly used in the industrial sector. The delta robot is constructed with three parallel arms, which can move in 3-DoF freely. Some delta robot configurations developed by researchers are parallelogram limbs [1-2], prismatic joints [3], and spherical magnetic joints [4]. The development of the delta robot needs to consider kinematics and the Graphical User Interface (GUI), besides construction. Both kinematics and the GUI must be developed to make it easy for the user to analyze the current robot states. The kinematics equation determines the relationship between the end effector pose and the robot joint actuator. The kinematics equation can be derived with different approaches. As presented in [5], they used the geometric method, while in [6], they used the DH model.

On the control system side, the researcher has conducted some research. Su et al. [7] controlled the tracking ability of the delta robot using PID, adaptive control, and sliding mode control. The experiment was carried out in simulation to investigate the effectiveness of the proposed controller. On the other hand, Azad et al. [8] performed the backstepping design to

ICAE 2022, October 05, Batam, Indonesia

DOI 10.4108/eai.5-10-2022.2327484

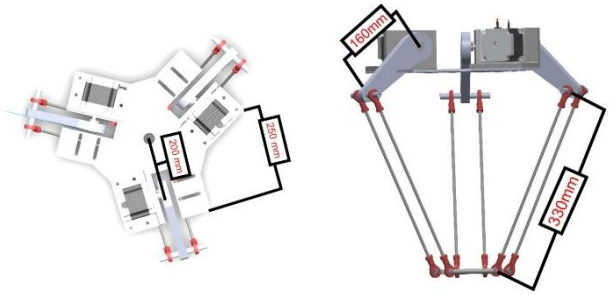


Fig. 1. The delta robot mechanical (a) top view, (b) the parallel arm joint.

Control a linear delta robot in simulation. In [9], they analyzed and simulated the solid work to get the exact motion process and then verified it with FANUC Robot M-2iA and ROBOGUIDE in a real-time application. While [10] they implemented the non-linear proportional derivative control (NPD) to overcome the delta robot tracking problem. They testified to the proposed method in parallel Delta-4 robots in real-time application [11]. An interface should be developed by employing the delta robot to move according to the motion command. In [12], Matlab software has been used to analyze kinematic movement. Moreover, the user can also control the joint actuator directly. Matlab was also used in [13] to calculate the forward and inverse kinematics. However, the simulation is performed using SolidWorks and used LabVIEW to create a user interface for the robot. This robot was developed as a virtual sorting machine. Some development of delta robot GUI currently only has a function to control joint actuator directly. However, the robot should be controlled by moving its end effector in the actual environment. In Computer Numerical Control (CNC) machines, the G-Code is commonly used to define the movement of the machine tip. The machine tip will move according to position coordinate and desired speed by giving a G-Code command. We adopt this system to develop the GUI and overcome prior research limitations [13]. Additionally, we used the C# and .NET frameworks as development tools for system design.

The rest of this paper is presented below. First, Section II discusses the proposed mechanical design of the delta robot, and then we elaborate inverse kinematics equation of the proposed design. Furthermore, we present the G-Code conversion and software design. Finally, the result and conclusion of this paper are presented in Sections III and IV.

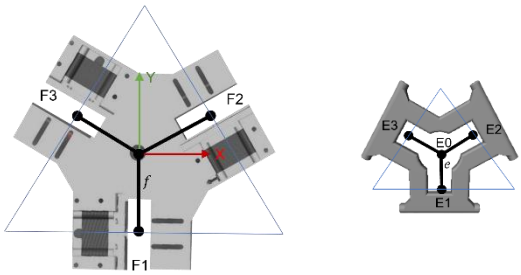
2 Mechanical Design and Method

2.1 Mechanical Design

The delta robot's mechanical design can be seen in **Fig. 1**. **Fig. 1** (a) represents the top view of the delta robot, and **Fig. 1** (b) illustrates the mechanical design fully. The motor is placed on the robot's top and arranged in a triangular position. The distance from the based robot center to the motor is about 20 cm, and the distance between each motor is around 25 cm. In terms of the arm, the length from the motor to the first joint is 16 cm and from the joint to the end effector is about 33 cm. The robot configuration can be seen in **Fig. 1** (b), and the actual delta robot's prototype is described in **Fig. 2**. **Fig. 2** (a) is the robot top view, and **Fig. 2** (b) is

(a) (b)

Fig. 2. The prototype of delta robot (a) top view, (b) the parallel arm joint.



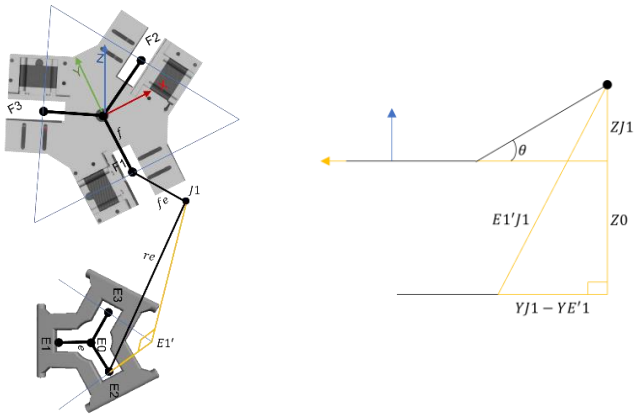
(a) (b)

Fig. 3. The delta robot joint configuration on (a) base robot, (b) end-effector.

The robot arm joint. In contrast to our previous work in [14], which used servo actuators and pneumatic grippers, we used 24 VDC stepper motors in this work. We made the main base from multiplex material and the short arm from 3D printed material. Furthermore, the end-effector gripper design can be seen in **Fig. 2** . We add the ball joint in between to connect the small arm to the long arm and ensure the robot can move in 3 DOF accordingly. This ball joint is illustrated by the green arrow in **Fig. 2** (b). Moreover, Arduino Uno has been chosen as the main controller to translate the G-Code command into actuator movement. We also used C# and .NET frameworks to design the GUI, translating G-Code into actuator movement.

2.2 Inverse Kinematics

We still adopt our previous Equation in [14] to obtain the robot inverse kinematics. The first step of obtaining the inverse kinematics is to arrange the parallel arm joint coordinate in cartesian as visualized in **Fig. 3**. **Fig. 3** (a) is related to the main base of the delta robot, and **Fig. 3** (b) denotes the robot end-effector. The F1-F3, which is described in **Fig. 3**, is denoted as the main base, while the E1-E3 illustrates the robot's end-effector, which is connected to the parallel robot arms. The f illustrates the distance from the main base center to the steeper motor shaft at the main base, and e is the length of each parallel arm to the end-effector.



(a) (b)

Fig. 4. The parallel arm position of F1 in (a) Y-Axis (b) X-axis.

When the arm is moved, then it can generate the right angle as presented on **Fig. 4**, we denoted this angle as $E1'$, beside $E1'$, and we also assumed that the other symbol which has been generated by this transformation, such as $J1$, the joint between robot body and the end-effector. Another, the fe and re also generated in this assumption. The fe denoted to the fix parallel arm's length from the stepper motor to the arm and re from the arm to the end-effector. In order to determine the length from $E1'$ to $J1$, we need to rotate the first motor to the Y-axis, as seen in **Fig. 4** (a). From this construction, we can determine the arm length aligned with the Y-axis using Equation (1). The $E0$ on **Fig. 4** (a) represented the center of the end effector and x_0, y_0, z_0 were the first step of each arm. Moreover, the $E1'$ was the result from $E1$ arm on the end-effector, $J1$ movement on Y-axis, and formed the triangle shape on the arm edge. Therefore, $E1'J1$ can be obtained by using a Pythagoras which was described in Equation (1).

$$\left.\begin{aligned} E0 &= (x_0, y_0, z_0) \\ E1 &= (x_0, y_0 - e, z_0) \\ E1' &= (0, y_0 - e, z_0) \end{aligned}\right\} E1E1' = x_0$$
$$E1'J1 = \sqrt{E1J1^2 - E1E1'^2} = \sqrt{re^2 - {x_0}^2} \quad (1)$$

After the $E1'J1$ has been determined, then we rotated the first arm to the X-axis, as presented in **Fig. 4** (b). The proposed of rotating the arm is to obtain the θ for the delta robot input movement. As presented in **Fig. 4** (b), we assumed two triangles were generated after rotating the arm on the X-axis. The θ that will be determined was located on the small triangle, yet we have to find the Y_{J1} and Z_{J1} beforehand. The Y_{J1} and Z_{J1} can be obtained by using Equation (2)-(3). If we substitute Equation (2)-(3) with Equation (1), then it can be transformed into Equation (4)-(5).

$$(Y_{J1} - Y_{F1})^2 + (Z_{J1} - Z_{F1})^2 = rf^2 \quad (2)$$
$$(Y_{J1} - Y_{E1'})^2 + (Z_{J1} - Z_{E1'})^2 = E1'J1^2 \quad (3)$$
$$(Y_{J1} - f)^2 + {Z_{J1}}^2 = rf^2 \quad (4)$$
$$(Y_{J1} - Y_0 + e)^2 + (Z_{J1} - Z_0)^2 = re^2 - {x_0}^2 \quad (5)$$

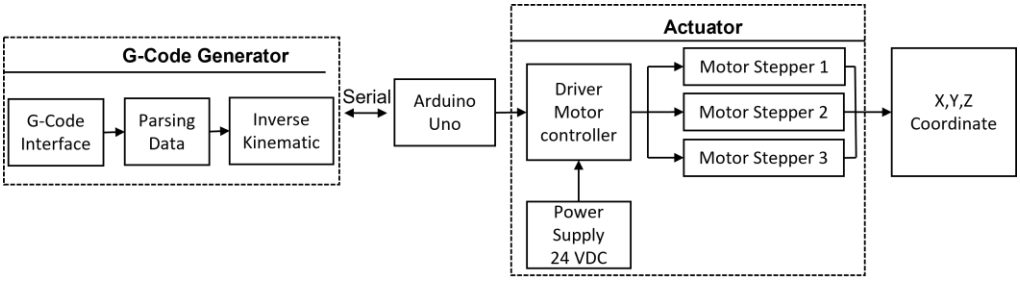


Fig. 5. The system block diagram of the G-Code interface.

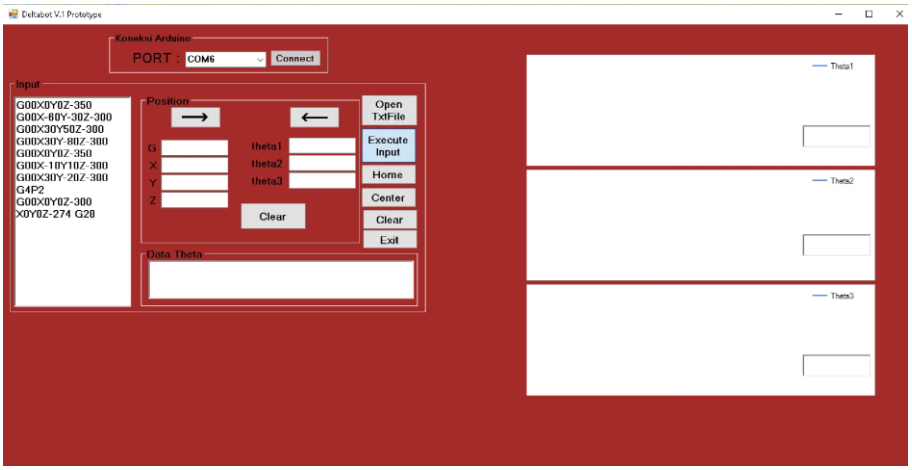


Fig. 6. The delta robot's Graphical User Interface (GUI).

After the $Y_{J1}$and Z_{J1} have been determined using Equation (2)-(3) or (4)-(5), then the θ can be represented under Equation (6). We assumed that the θ represents the E_0 coordinate at the end-effector. Therefore coordinate of the end-effector when it moved can be determined by Equation (7). Because the angle generated from the parallel arm was about 120°, we used this angle to determine the coordinate at the end-effector as presented in Equation (7).

$$\theta = \arctan\left(\frac{z_{J1}}{y_{F1}-y_{J1}}\right) \tag{6}$$
$$\theta \rightarrow E_0\ (x_0, y_0, z_0)$$

$$E_0'(x_0', y_0', z_0') \begin{cases} x_0' = x\cos(\pm 120°) + y\sin(\pm 120°) \\ y_0' = -x\sin(\pm 120°) + y\cos(\pm 120°) \\ z_0' = z_0 \end{cases} \tag{7}$$

2.3 System Block Diagram

The system block diagram of the G-Code interface is represented in **Fig. 5**. This system consists of two main systems: the G-Code generator and the actuator system. On the G-Code generator, as presented in **Fig. 5**, we developed the interface using the C# programming

language, where the GUI can be seen in **Fig. 6**. As we can see in **Fig. 6**, our interface has the input G-Code to pop up the command from the text file, and then we need to click the Execute Input to execute this command. Once the Execute Input has been pushed, the θ of each arm will appear in the Data Theta text box. On the G-Code interface, we used the general command to move the delta robot according to the coordinates given to the robot. For example, sending the code G00X-150Y-15Z- 350 means that the robot will move to coordinate -150 on the X-axis, -15 on the Y-axis, and -350 on Z-axis. Moreover, this code will be parsed in the parsing data process to collect the coordinate data. The coordinate data at the input of the end effector will be used for the inverse kinematics. After the system calculates the kinematics result, we send the data to Arduino Uno through serial communication. The Arduino Uno will convert the angle data into electric pulses so the drive motor can understand the command. For activating the motor driver, we used the 24 VDC as the power source, and this motor driver will send the pulse to each stepper motor to move accordingly with the input coordinate from the G-Code interface.

3 Experimental Results

In order to verify our system in real-time, the experiment has been done in this section by ordered the robot to move accordingly to the G-Code command, including testing the G-Code interface to the robot movement and the robot movement separately. As presented in **Fig. 6**. In **Fig. 6**, we send in about ten commands from our G-Code interface to make the robot move accordingly with the input. And then, let the system generate the θ for each arm according to the input command from the G-Code.

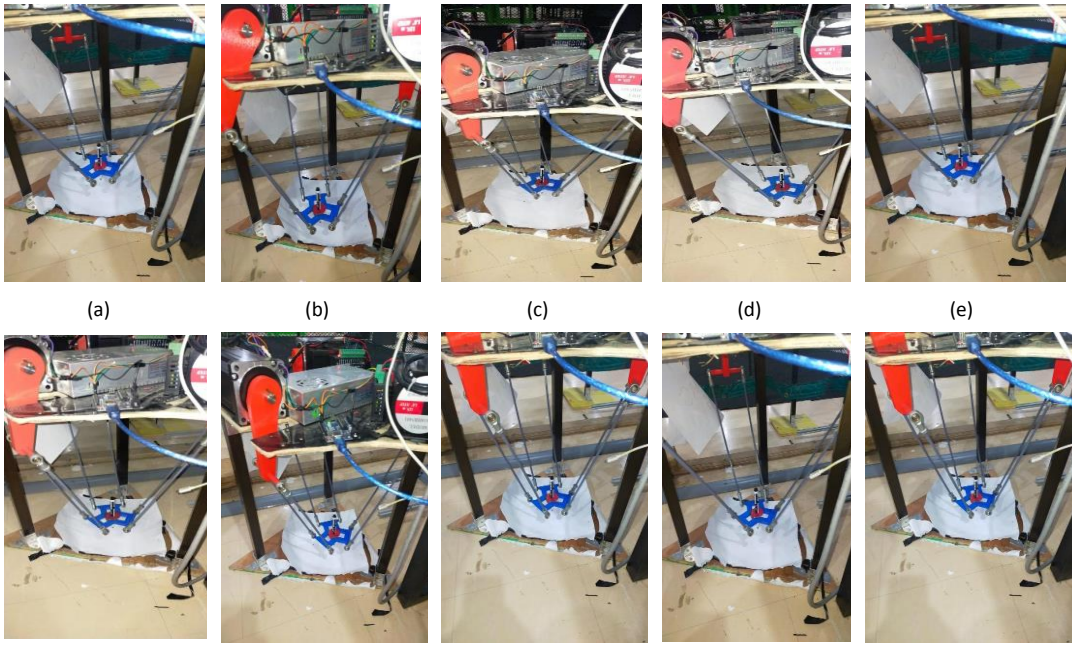


Fig. 7. The movement of the delta robot according to the G-Code command.

The results of this movement are represented in **Fig. 7**, where **Fig. 7** (a) was the movement of command **G00X0Y-Z-350**, this command let the robot move to coordinate (0,0,-350), which is the initial coordinate before moving to another command. On **Fig. 7** (b) the robot moved to command **G0X-60Y-30Z-300**, which was in the coordinate (-60, -30, -300). As for **Fig. 7** (c), the robot moved to coordinate (30,50,-300), which was the position result from command **G0X30Y50Z**-300. The next step, which is illustrated on **Fig. 7** (d), describes the movement of the delta robot for **G0X30Y-80Z-300** allowed the robot to move in coordinate (30, -80, -300) and **Fig. 7** (e) **G00X0Y-Z-350** in the coordinate (0,0, -350), this movement allowed the robot to went back to its original position to do next movement. After the robot went back to its original position, then on **Fig. 7** (f), we commanded to move to coordinate (-10, 10, -300) with the G-Code command represented as **G0X-10Y10Z-300**, then moved to coordinate (30, -20, -300) as presented on **Fig. 7** (g) for command **G00X30Y-20Z-300**. Furthermore, the rest of the picture represented **G4P2 Fig. 7** (h) where this command meant to ask the robot to delay its movement in about 2 seconds, and went back to its original position with command **G00X0Y-Z-300** as presented **Fig. 7** (i), and **Fig. 7** (j) for command **G28** to stop the robot movement. In order to understand the result of this movement, a permanent marker has been attached to the robot's end-effector. The marked result from this experiment can be seen in **Fig. 8**, which was the "A" letter.

While the robot moves accordingly with the G-Code input, the interface result during the movement of making the letter "A", the angle of each joint has been recorded in this interface. The θ of each joint presented in **Fig. 9** will be changed accordingly with the G-Code command, the change was illustrated by the graph on **Fig. 9**. As for the θ of each arm joint which was generated by the kinematic, represented in **Table I**. The result of **Table I** showed that each coordinate could transform into an angle as input for the Arduino, in the case of making letter "A", the 10 commands were needed by the robot. The interesting point is that the commands **G00X30Y-20Z-300** and **G4P2** produced the same angle for each parallel arm. This is because the command **G4P2** allows the robot to delay the movement by about 2 seconds, so the angle is still the same as the previous one.

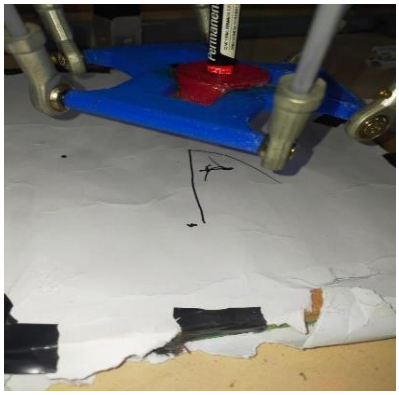


Fig. 8. The result of the G-Code command.

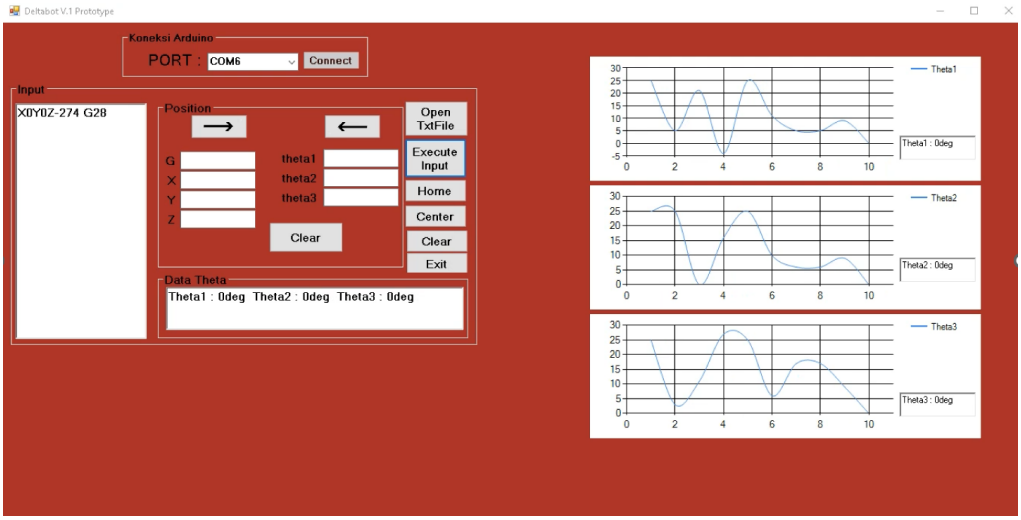


Fig. 9. The G-Code interface while running the command.

Table. I. The result of θ angle of each joint is generated from the coordinate.

Command	End-effector Coordinate (mm)			The angle of θ (°)		
	X	***Y***	***Z***	***θ1***	***θ2***	***θ3***
G00X0Y-Z-350	0	0	-350	25	25	25
G0X-60Y-30Z-300	-60	-30	-300	5	-15	3
G0X30Y50Z-300	30	50	-300	21	0	11
G0X30Y-80Z-300	30	-80	-300	-4	16	27
G00X0Y-Z-350	0	0	-350	25	25	25
G0X-10Y10Z-300	-10	10	-300	11	10	6
G00X30Y-20Z-300	30	-20	-300	5	6	17
G4P2	30	-20	-300	5	6	17
G00X0Y-Z-300	0	0	-300	9	9	9
G28	0	0	-274	0	0	0

4 Conclusion

This paper presented the design of a delta robot and a simple G-Code interface under C# programming and implemented in real-time application. In order to generate the movement of the delta robot, we used the same kinematic Equation as our previous work. As we can see in the experiment results, our G-Code interface can send the command, and the delta robot's kinematic can understand the command. Each θ of the parallel arm can also be determined in real-time application. In the future, we will consider adding a vision system to detect the object's color and coordinate so that this robot can be used as the pick and place instrument.

References

[1] R. Clavel, "Device for the movement and positioning of an element in space," U.S. Patent 4 976 582, Dec. 11, 1990.

[2] Li, Y., Shang, D., & Liu, Y. (2019). Kinematic modeling and error analysis of Delta robot considering parallelism error. International Journal of Advanced Robotic Systems.

[3] B. Mehrafrooz, M. Mohammadi and M. T. Masouleh, "Kinematic Sensitivity Evaluation of Revolute and Prismatic 3-DOF Delta Robots," 2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM), 2017, pp. 225-231, doi: 10.1109/ICRoM.2017.8466159.

[4] S. Ahangar, M. V. Mehrabani, A. Pouransari Shorijeh and M. T. Masouleh, "Design a 3-DOF Delta Parallel Robot by One Degree Redundancy along the Conveyor Axis, A Novel Automation Approach," 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI), 2019, pp. 413-418, doi: 10.1109/KBEI.2019.8734975.

[5] C. Liu, G. -H. Cao and Y. -Y. Qu, "Workspace Analysis of Delta Robot Based on Forward Kinematics Solution," 2019 3rd International Conference on Robotics and Automation Sciences (ICRAS), 2019, pp. 1-5, doi: 10.1109/ICRAS.2019.8808987.

[6] Yong-Lin Kuo, Mathematical modeling and analysis of the Delta robot with flexible links, Computers & Mathematics with Applications, Volume 71, Issue 10, 2016, Pages 1973-1989, ISSN 0898-1221, https://doi.org/10.1016/j.camwa.2016.03.018.

[7] T. Su, X. Liang, G. He, Q. Zhao and L. Zhao, "Robust Trajectory Tracking of Delta Parallel Robot Using Sliding Mode Control," 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 508-512, doi: 10.1109/SSCI44817.2019.9003125.

[8] F. A. Azad, S. Rahimi, M. R. Hairi Yazdi and M. T. Masouleh, "Design and Evaluation of Adaptive and Sliding Mode Control for a 3-DOF Delta Parallel Robot," 2020 28th Iranian Conference on Electrical Engineering (ICEE), 2020, pp. 1-7, doi: 10.1109/ICEE50131.2020.9261040.

[9] Y. Zhi-Xiang, J. -Q. Peng and M. -Y. Chen, "Adaptive Control of a Linear Delta Robot Based on Backstepping Design," 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), 2018, pp. 1138-1141, doi: 10.1109/SCIS-ISIS.2018.00179.

[10] H. Zhang, K. Zhang and J. Gao, "Dynamically Based Motor Parameters for Delta Robots Using the Specified Workspace," 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA), 2019, pp. 814-818, doi: 10.1109/IEA.2019.8715064.

[11] C. E. Boudjedir, D. Boukhetala and M. Bouri, "Non-linear PD control of a Parallel Delta robot: Expermentals Results," 2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM), 2018, pp. 1-4, doi: 10.1109/CISTEM.2018.8613618.

[12] Rodríguez-Franco, Martín & Jara-Ruiz, Ricardo & López-Álvarez, Yadira & Rodríguez, Juan Carlos. (2021). "Development of interface for kinematic analysis of a delta-type parallel robot". Journal of Computational Systems and ICTs. 18-27. 10.35429/JCSI.2021.20.7.18.27.

[13] Huang, Xiaoping & Weng, Fangyi & Wei, Zhongxin & Kamruzzaman, M.M.. (2021). Application and research of robot sorting system based on LabVIEW. Journal of Intelligent & Fuzzy Systems. 1-8. 10.3233/JIFS-219090.

[14] Susanto, M. Rohim and R. Analia, "The Implementation of a Modest Kinematic Solving for Delta Robot," 2019 2nd International Conference on Applied Engineering (ICAE), 2019, pp. 1-5, doi: 10.1109/ICAE47758.2019.9221720.