Scientific Journal Widya Teknik
Volume 22 No. 1 2023
ISSN 1412-7350 eISSN 2621-362

DESIGN AND IMPLEMENTATION OF POSE RECORDING SYSTEM WITH DENAVIT HARTENBERG METHOD IN A 6-DOF ROBOT ROTARIC

Calvin Susanto[1*], Fransisko Limanuel[2], Ferry Rippun[3]
[1]Electrical and Electronics Engineering Faculty
Atma Jaya Catholic University of Indonesia
Jakarta, Indonesia 15345
*e-mail : [1]calvinsusreal@gmail.com

ABSTRACT

Robot developments have been integrated into several industrial and home appliances. To keep up with the development, UNIKA Atma Jaya created a group called PATRIOT as a head start, and started the development on ROTARIC robot. One of the early robot research discussed in this paper is to design and implement ROTARIC 6-DOF robot pose recording system which is drawn using forward kinematics with Denavit Hartenberg method. The system has a GUI with several function inside such as robot pose recording, 2-D display of the robot recorded pose using Forward Kinematics analysis, and playback recorded pose. The system uses several hardware that have been connected serially such as 6 Dynamixel MX-28 servo, OpenCM 9.04 +485 EXP microcontroller, and a Computer. The Software used in the system is C# .NET with WinForms App as the GUI and Arduino IDE. From the test done in this research, the recording and 2-D drawing system achieved error <±1cm or 6.67% relative error caused by the resolution value while reading the servo position sensor and the value rounding inside the program. Meanwhile, the playback recording function that uses queue system achieved error of <±1cm and error of ±5cm or 33.3% relative error when the recorded pose is played using real time delay used in recording. The error is caused by the robot failure to keep up with the speed of the recorded pose and the limited torque in servo number 2 in the robot.

ABSTRAK

Perkembangan robot telah diintegrasikan kedalam berbagai peralatan industry dan rumah. Untuk menyamakan perkembangan tersebut UNIKA Atma Jaya membuat grup bernama PATRIOT sebagai tumpuan awal dan telah memulai pengembangan untuk robot ROTARIC. Salah satu penelitian yang didiskusikan di paper ini adalah desain dan implementasi sistem perekaman pose robot ROTARIC 6-DOF yang digambarkan menggunakan forward kinematic dengan metode Denavit Hartenberg. Sistem tersebut memiliki GUI dengan berberapa fungsi didalamnya seperti perekaman pose robot, penggambaran pose robot 2-D menggunakan analisis Forward Kinematik dan pemutaran pose rekaman. Sistem tersebut menggunakan berberapa perangkat keras yang dihubungkan secara serial seperti 6 buah servo Dynamixel MX-28, mikrokontroller OpenCM 9.04 +EXP 485, dan sebuah Komputer. Perangkat lunak yang digunakan pada sistem yaitu C# .NET dengan WinForms App sebagai GUI dan juga Arduino IDE. Dari pengujian yang dilakukan pada penelitian ini, sistem perekaman dan penggambaran 2-D memeperoleh nilai error <±1cm atau relative error sebesar 6.67% disebabkan oleh nilai resolusi pada saat pembacaan sensor posisi pada servo dan juga pembulatan didalam program. Sementara, pada fungsi pemutaran rekaman yang menggunakan sistem antrian memperoleh error <±1cm dan untuk pemutaran rekaman yang menggunakan waktu jeda meperoleh error ±5cm atau relative error sebesar 33.3%. Error tersebut disebabkan oleh kegagalan robot untuk menyamakan kecepatan dengan kecepatan rekaman pose, dan juga keterbatasan torsi pada servo kedua pada robot.

Keywords: *6-DOF Articulated Robot; Robot pose recording; Forward Kinematics; Dynamixel MX-28; OpenCM 9.04.*

I. Introduction

Robotics technologies have advanced and integrated in every form of electronic device from several home appliances to industrial scaled automated manufacturers. Robots services is to replace direct human work automatically, as efficient and reliably repeatable[1]. Faculty of Engineering of UNIKA Atma Jaya have started a team called “Pusat Robotik dan Internet of Things” (PATRIOT). The team intent is to develop an articulated robot manipulator as a

head start of robotics research in UNIKA Atma Jaya.

Early development stage of this robot is to design and build robotic arm with 6-Degree of Freedom (DOF) called ROTARIC (Rotary Articulated Arm Robot with Interchangeable Component). The robot is intended to have features such as easy to modify, movement position recording, fast response, accurate and User Interface capable of controlling and monitors through programs. This development is expected to bring the basic research in robotic design, programming, and functionality that may led to more advanced research in the future.

This works dives into the recording functionality in the robotic arm. The recording function aims to make controlling and programming the robot easier through input based on where the links and joints condition in real time by manually moving the robot by hand. Then after the recorded links and joints have been saved, the robot can be moved according to the recorded information. Secondly the recorded information can be displayed as a movement simulation using a user interface.

II. Theory/ basic theory

II.1. Articulated Robotic Arm

The articulated robotic arm is the type of robot manipulator where several robot links are connected serially by either a revolute or prismatic joint from the base of the robot to the end-effector[2]. Because of this configuration the robot has many poses based on its joint condition that can also change its end-effector position and orientation[3]. To determine the position of the joints and link in a Coordinate space uses a solution called Forward Kinematics[4].

II.2. Forward Kinematics with Denavit Hartenberg Parameters

In Robotics application, Forward Kinematics can be used to determine a robot coordinates in a 3-D plane based on the robot joint and links[5]. There are many forward kinematics models to do this. One of which is the Denavit-Hartenberg(D-H) model that is the most common one to use[6].

To determine the pose of a robotic arm in a Cartesian coordinate, the first step is to assign the Denavit Hartenberg parameter on each joint based on the position and orientation of the joint. Before the Denavit Hartenberg is assigned, each joint is attached with 3-D coordinate frame by following some prerequisite/rules[7].

1. Each joint axis either rotational or linear must be assigned as axis Z_i.
2. Define the axis X_i perpendicular to axis Z_i and axis Z_{i-1}.If possible, define X_i parallel to X_{i-1}.
3. Define axis Y_i perpendicular to both Z_i and X_i according to right hand rule.

After each joint orientation and position have been determined, the next step is to determine the D-H parameters of each joint. The D-H parameters of a joint consist of 4 parameters[7]:

1. Parameter d is the offset from the center of previous joint O_{i-1} to the center of current joint O_{i-1} measured from Z_{i-1}.
2. Parameter θ (theta) is the angle needed to rotate the joint around Z_{i-1} until axis X_{i-1} is aligned to X_i.
3. Parameter a is the offset from Oi-1 to Oi measured from axis Xi
4. Parameter α (alpha) is the angle needed to rotate the joint, around axis X_i until axis Z_i is aligned to Z_{i-1}

After the D-H parameter is determined then the next step is to create homogenous transformation matrix based on the D-H parameter. The matrix consists of 2 part, the 3x3 matrix rotational part and 3x1 matrix translation part as pictured in equation 1.

$$[T] = \begin{pmatrix} \begin{vmatrix} & R & \\ & & \\ 3 & x & 3 \end{vmatrix} & \begin{vmatrix} T \\ \\ 3x1 \end{vmatrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{pmatrix} \quad (1)$$

Homogenous matrix of Denavit Hartenberg is the transformation matrix of a joint that consist of 2 components. First, is the transformation along the Z_i axis on the joint depicted as 4x4 matrix $[Z_i]$ which is the result of translation along axis Z_i by d_i $[T_{Zi}(d_i)]$ and rotation around axis Z_i by θ_i $[R_{Zi}(\theta_i)]$.

$$[T_{Zi}(d_i)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & di \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$[R_{Zi}(\theta_i)] = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$[Z_i] = [T_{Zi}(d_i)][R_{Zi}(\theta_i)] \quad (4)$$

Second, is the transformation along axis X_i on the joint depicted as 4x4 matrix $[X_i]$ which is the result of translation along X_i by a_i $[T_{xi}(r_i)]$and rotation around X_i by α_i $[R_{xi}(r_i)]$.

$$[T_{Xi}(a_i)] = \begin{bmatrix} 1 & 0 & 0 & ai \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$[R_{Zi}(\alpha_i)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$[X_i] = [T_{Xi}(a_i)][R_{Zi}(\alpha_i)] \quad (7)$$

The two 2 component is multiplied as equation 8 and 9 which resulted in 4x4 matrix $[{}^{m}_{n}T]$ indicating a transformation from joint m to joint n.

$$[{}^{m}_{n}T] = [Z_i][X_i] \quad (8)$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\cos\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & dn \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

After the matrix of each DH-Parameter have been created, then the next step is to multiply each transformation matrix from the base $[{}^{0}_{1}T]$ with the next matrix $[{}^{1}_{2}T]$ which created the matrix result $[{}^{0}_{2}T]$. Then the current matrix result is multiplied by the next transformation matrix $[{}^{2}_{3}T]$ which created the next matrix result $[{}^{0}_{3}T]$ and so on until the n transformation matrix has been multiplied which created the final result matrix $[{}^{0}_{n}T]$ as depicted in equation 10.

$$[{}^{0}_{n}T] = [{}^{0}_{1}T][{}^{1}_{2}T] \dots [{}^{n-1}_{n}T] \quad (10)$$

After each calculation is done the position of each joint (P_x,P_y,P_z) can be acquired by extracting the 3x1 matrix on the upper right corner from each end-effector matrix ($[{}^{0}_{n}P]$) depicted below in equation 11.

$$[{}^{0}_{n}T] = \begin{pmatrix} & & & \\ & R & & \begin{vmatrix} P_x \\ P_y \\ P_z \end{vmatrix} \rightarrow {}^{0}_{n}P \\ 3 & x & 3 & \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (11)$$

III. Research methodology

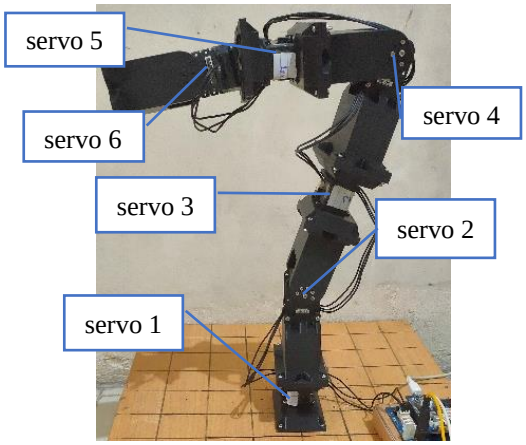

Figure 1. ROTARIC Robot Structure Design

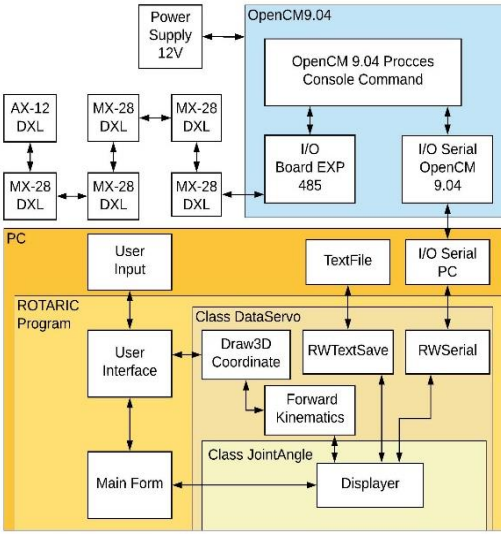

Figure 2. ROTARIC System Diagram

III.1. Robot Structure and System

The ROTARIC robot is an articulated robotic arm type. On this research the robot is a 6 DOF robotic arm ,with the structures are shown in figure 1. System of the 6 DOF robotic arm is designed as block diagram on figure 2.

Six Dynamixel MX-28[8] servo is used as rotational joint between the links connected serially. Then the robot is connected to an OpenCM 9.04[9] and its expansion board EXP 485[10] microcontroller as a serial interface to the robot and connected through serial interface on the computer. The computer has a Visual C# .Net program that processing all the input or output from the User Interface and calculating the kinematics of the robot. The C# program uses Object Oriented Programming divided into 5 classes by its role in the program. The UI interface is displayed through figure 3 to 5.

The Recording system is started by giving an input(in this case a press of a button to start the function, and textbox time delay between each records) to the UI in the C# program. Then the program sends the command to get each servo angle to the OpenCM 9.04 through the serial port and then the OpenCM9.04 sends the command to

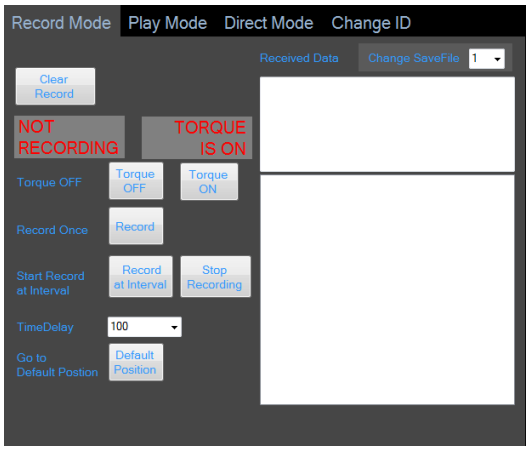

Figure 3. Record Mode UI

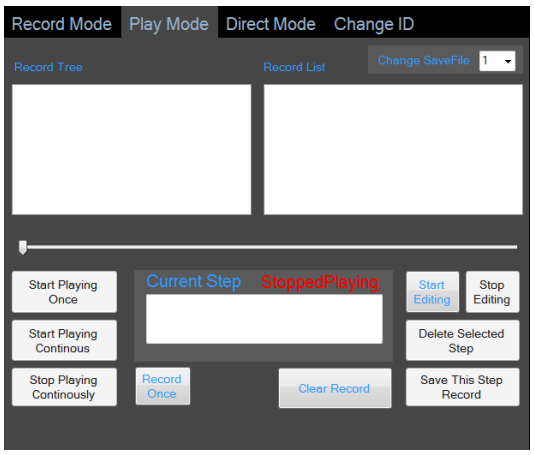

Figure 6. Play mode UI

read the register where the current servo angle is saved through its serial port to the servos.

Then the servo writes the angle data back to the serial port to the OpenCM 9.04. Then the servos angle is sent back from OpenCM serial port to the C# Program on the PC. The angle then saved in a text file according to the chosen save file , including the time delay used for the recording. Then the program waits until either stop recording command is called or until the time delay is up which then the program is looped back to the start of recording. While this recording program is looping, the robotic arm then can be turned manually by hand and the pose can be saved at when the time delay is up.

Once the recording is saved and the recording program is stopped then recorded pose can be played back. First the playback program read the saved angle and then send the command

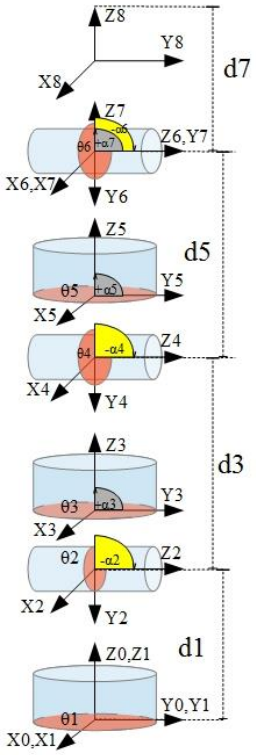

Figure 5. Robot Coordinate Frames

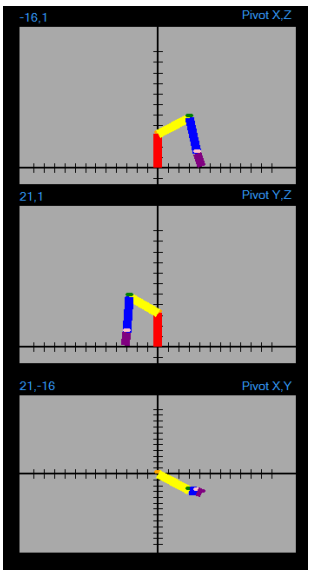

Figure 4. The 3-D representation with 2-D plane Pivot

to write the angle to the OpenCM 9.04. Then the OpenCM 9.04 writes the position to the servos as goal position to turn to. Meanwhile, another program is used to determine and draw the coordinates of each servo in a 3-D space.

To determine the robot coordinates in a 3-D plane, requires a method to calculate the position and orientation, which will be the Forward Kinematics.

III.2. Forward Kinematics

Based on its structure, the robot joints frame and orientation can be described by putting 8 frames on the robot as pictured on figure 6. Then based on the joint frames, the DH-Parameter can be created on table 1. Based on the joint frames, the robot only has the parameter θ_i that changed dynamically, the other parameters are statics parameter.

Based on the D-H parameter on table 1, the D-H matrix $[{}^{n-1}_{n}T]$ is created from $[{}^{0}_{1}T]$ to $[{}^{7}_{8}T]$.Multiplying each matrix sequentially from $[{}^{0}_{1}T]$ to $[{}^{6}_{7}T]$ resulted in each end-effector matrices for each joint until the last end-effector matrix $[{}^{0}_{7}T]$ as depicted in equation 12 through 18.

Table 1. D-H Parameters

i	a	α	d	θ
1	0	0	0	0
2	0	-90	d1	θ1
3	0	90	0	θ2
4	0	-90	d3	θ3
5	0	90	0	θ4
6	0	-90	d5	θ5
7	0	90	0	θ6
8	0	0	d7	0

$$[{}^{0}_{2}\mathrm{T}] = [{}^{0}_{1}\mathrm{T}][{}^{1}_{2}\mathrm{T}] \quad (12)$$

$$[{}^{0}_{3}\mathrm{T}] = [{}^{0}_{2}\mathrm{T}][{}^{2}_{3}\mathrm{T}] \quad (13)$$

$$[{}^{0}_{4}\mathrm{T}] = [{}^{0}_{3}\mathrm{T}][{}^{3}_{4}\mathrm{T}] \quad (14)$$

$$[{}^{0}_{5}\mathrm{T}] = [{}^{0}_{4}\mathrm{T}][{}^{4}_{5}\mathrm{T}] \quad (15)$$

$$[{}^{0}_{6}\mathrm{T}] = [{}^{0}_{5}\mathrm{T}][{}^{5}_{6}\mathrm{T}] \quad (16)$$

$$[{}^{0}_{7}\mathrm{T}] = [{}^{0}_{6}\mathrm{T}][{}^{6}_{7}\mathrm{T}] \quad (17)$$

$$[{}^{0}_{8}\mathrm{T}] = [{}^{0}_{7}\mathrm{T}][{}^{7}_{8}\mathrm{T}] \quad (18)$$

This method from the start of defining D-H parameter until getting each position of end-effector is repeated every time the drawing function gets called using each servo angles as an input parameter.

IV. Research results and discussions

IV.1. Servo MX-28Torque Test

The servo needs to be tested for servo torque to determine the availability of torque. This test is done by several step.

1. The servo is put on a fixed base about 1.5 meter above the floor.
2. Changed the servo register Moving_Speed to a variable value (100,200,500). Changed the servo angle from 0° (downward) to 90° (sideward).
3. A steel rod about 20 cm is mounted on the servo horn. The rod weight is 55 grams. The steel rod is marked with tape at 5 cm, 10 cm, 15 cm and 20 cm.
4. Then a string is attached at the first mark. A bottle with water as variable weight is attached to the string. The bottle was made to be hanging above the floor.
5. Change the servo angle to 180°(upward) slowly(e.g. 10° in difference), if it able to hold the weight and not sending an alarm message.
6. The torque is calculated by multiplying weight(m), gravity(g) acceleration, and rod length(l) as displayed in equation 19.

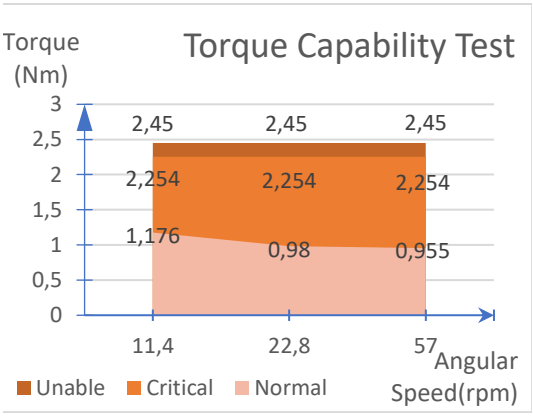

Figure 7. Torque Capability Test Graph

$$\tau = m.g.l \quad (19)$$

7. If the servo was able to hold and moved, the current variable weight and rod length is marked with an “normal” mark.
8. If the servo was able only to hold or could be moved to certain angle before sending alarm message, the current variable weight and rod length is marked with a “critical” mark.
9. If the servo was not able to neither hold nor move, the current variable weight and rod length is marked with a “unable” mark.
10. The test is repeated by changing its weight, rod length, and servo speed until certain rod length received 2 “unable” marks in a row.

The test result is displayed in graphics on figure 7. The result above is tested for Moving_Speed 100 or 11.4 rpm, 200 or 22.8 rpm, and 500 or 57 rpm.

According to this test result the servo can move 0 -180° and have the torque available to hold for the appropriate torque caused by gravity about 1,176 Nm for Moving Speed 100, 0,98 Nm for Moving Speed 200, 0,882 Nm for 500 Moving Speed. The Torque marked with “critical” is the critical point where the servo is not able to withstand the torque while moving but it can hold the weight using its stall torque. The Torque marked “unable” is the point where the servo unable to withstand the arm weight using its stall torque.

The result test is compared to the datasheet of the servo MX-28[8]. Based on the datasheet, the stall torque of the servo is 2.5Nm at 12V and using speed of 11.4 rpm can result about 1.176Nm based on the datasheet performance graph. This results in 0.05 error difference in stall torque, and 0 moving torque between the test and datasheet. The error can be caused by the hardware specification, especially the power supply voltage and current used. But the error result is a minimized error on stall torque

According to this torque test, current robot structure and construction is torque limited mostly at servo 2, which hold the longest axis from joint 2 to the endpoint about 0.45 m in length and weights 0.7 kg. This resulted in downward Torque around 1.54Nm that cannot be handled by the servo MX-28 while moving. The torque limit prevents the servo from moving and sometimes can overload the stall torque. As a result of this problem, the resultant length from the axis of joint 2 to the end point should not exceed 0.3 meter and uses the 11.4 rpm as the angular speed.

IV.2. Servo MX-28 Angle Accuracy Test

This test is to determine the accuracy of the servo angle. The servo is instructed to rotate at variable angle (0°-360°), the angle then converted into servo angle (0-4095), and then after the servo reach the position, the servo is instructed to read its current angle and displayed it while the

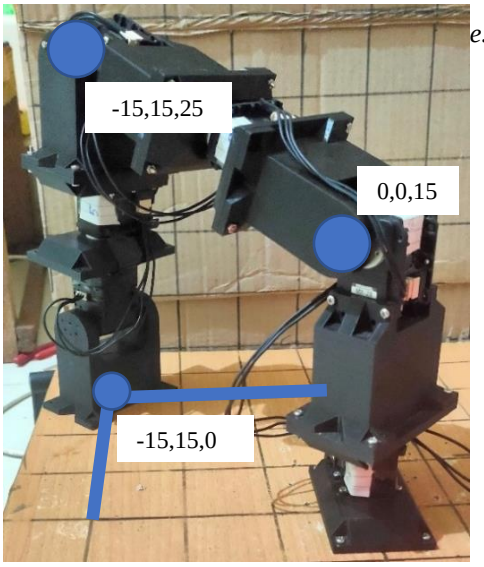

Figure 11. Pivot real pose 1

servo
is

measured by a protractor with accuracy of 1°. The result is detailed in table 2.

The angle test, results in no difference in measurement using the protractor. There are differences in measurements using the position sensor inside the MX-28 Servo. The position error creates about ± 2 servo angle which resulted in about ± 0.176° difference when displaying the joint angle, but other than that it is very minimalized error.

IV.3. Record and Display Program Test

The record program is tested by manually turning the robot from a position to another 2 different position, while the record program is running. The robot is put into a 3-side box, with each side representing plane XZ,YZ, and XY with a tick marks of 5 cm for each axis. Pictures are taken when the robot at its first pose, and anoth the next pose, and ot pose. Then the isplayed plane on program resulted i h servo.

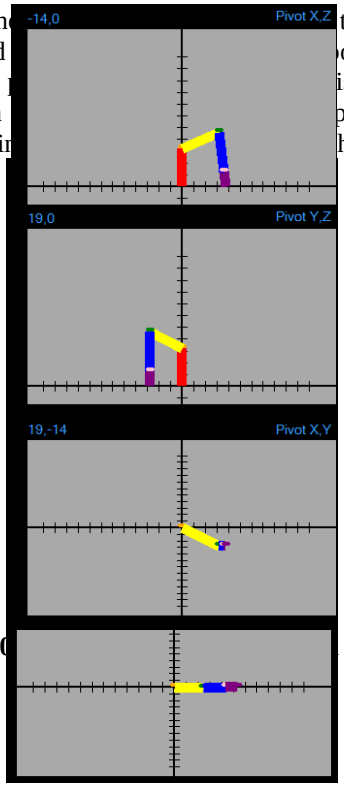

Figure 10 XY of

Figure 9. Displayed pivot of pose 2

Table 2. Angle Accuracy Test Result

Angle	Servo Angle	Read Angle	Measured Angle
0	0	0	0
36	410	410	36
72	819	820	72
90	1024	1024	90
108	1229	1228	108
144	1638	1637	144
180	2048	2047	180
216	2457	2458	216
252	2867	2869	252
270	3071	3072	270
288	3276	3276	288
324	3686	3687	324
360	4095	4094	360

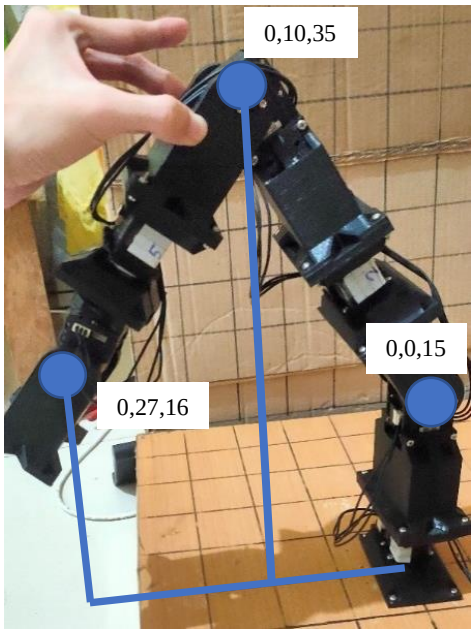

Figure 8. Pivot real pose 2

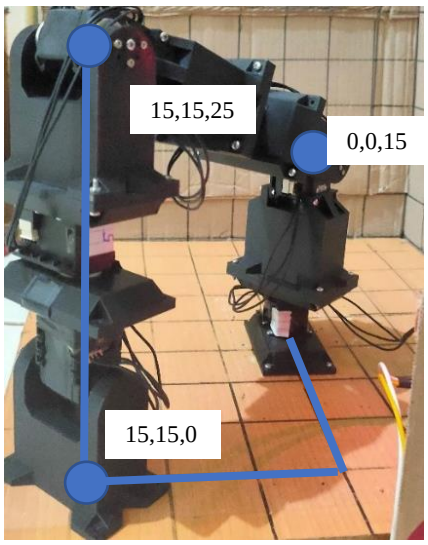

Figure 12. Pivot real pose 3

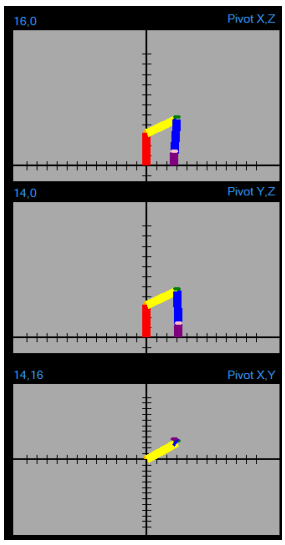

Figure 13. Displayed pivot of pose 3

The pose is displayed on figure 8,10, 12. The 2D plane of each pose are displayed in figure 9, 11 and 13.

The Record Program test is done successfully at given pose with minimum XYZ position errors. The errors made between the real angle and the displayed one, possibly caused by 3 issues, first is the amount of readable resolution of the servo position encoder, second is the rounding of the angle data inside the program, third is the misalignment made in the current robot structure and model that could have changed the position of each joint and links,. But this error is minimalized to ± 1cm ors relative error of 6.67%.

IV.4. Play Record Program Test

The Play Record Program is tested by commanding the Play program to play the recorded data that already done in Record Program test, then the last pose of play record program is compared with the recorded last pose. This test is done in two different way, each done in 5 times. Both of the test is done using the Box scale of 5cm as the measurement tool. The first test is done by waiting until each servo has arrived at the current position before giving the next position.

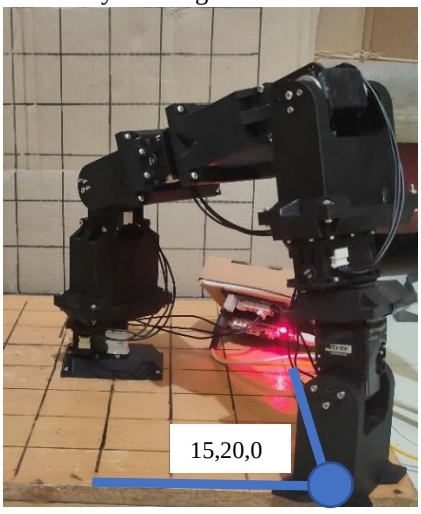

Figure 14. Robot pose after playing the record

The test result achieved almost the same as figure 12, displayed in table 3, with error of <±1cm or 6.67% Relative Error. This means when the robot is using the queue system in its playback program, the result position should be the same as when the robot poses recorded. The second test is done by using the time delay recorded as the time delay. The result of the last pose is displayed in figure 14. The full result of the test is displayed on table 3. After comparing the result of the robot during recording, with the last pose after the recording is played, an error of ±5cm or relative error 33.3% occurred. This error is caused by three problems.

First problems can be caused by the pose that is saved during recording, which could be too fast for the servo to keep up with. Especially when the servo pose is recorded manually by hands without knowing the real time speed. Second, is the limited torque of the servo especially servo number 2, which causing the speed of this servo to be lower than expected 11.4 rpm.

When the servo is turned manually, the torque requirements to move the robot to certain position is fulfilled by the torque created by the operator hand and gears inside the servo motor. When the robot is playing the recorded pose, the

Table 3. Play Record Test Result

No	Play Record Mode											
	Queued Error (cm)						Time delay error (cm)					
	x	y	z	r	Ae	Re	x	y	z	r	Ae	Re
1	0	0	0	0	0	0	0	5	0	5	5	33.3%
2	0	0	0	0	0	0	0	5	0	5	5	33.3%
3	0	0	0	0	0	0	0	5	0	5	5	33.3%
4	0	0	0	0	0	0	0	5	0	5	5	33.3%
5	0	0	0	0	0	0	0	5	0	5	5	33.3%

torque required to move is only created by the servo. This in turn causing the robot pose at certain time, different than the recorded pose at that exact time when playing the recorded pose. And when given the next pose, the robot, with its already wrong position is causing the next pose to be wrong as well. This sequence of error can cause accumulated error from one pose to another until the last pose is given. And this problem can cause different problem as well, such as 2 links crossing each other while the servo is turning to its goal position (which may break the links), or making the robot pushing the bottom plane like the example case in this test.

The third problem is the misaligned angle created during recording caused by the robot structure.

V. Conclusion

This research has developed several ROTARIC program functions including the recording, play record and 2-D display using forward kinematics. The Record mode and 2-D display uses Denavit Hartenberg method to calculate the Forward Kinematics solution and has achieved minimum errors of <±1 cm or 6.67% relative error difference from the real time position. Meanwhile, the playback recording function that uses queue system achieved error of <±1cm or of 6.67% relative error and error of ±5cm or 33.3% relative error when the recorded pose is played using real time delay used in recording after moving to 2 recorded poses as a result of its servo number 2 limited torque capability. In the next development, further improvement can be made to reduce these errors either by changing the servo 2 torque capability to a different servo type, improvement on robot structure, or better structure on the program to handle the recording and playback system accurately.

Acknowledgement

This research was supported by Universitas Katolik Indonesia Atma Jaya funded by Fakultas Teknik Unika Atma Jaya

References

1. Pitowarno. Endra. 2006. ROBOTIKA: Desain, Kontrol, dan Kecerdasan Buatan. Andi Offset. Yogyakarta:
2. Gerald Cook. 2011. Mobile Robots: Navigation, Control and Remote Sensing. Wiley-IEEE Press.
3. Ovy, Enaiyat Ghani & Ferdous, S.M. & Rokonuzzaman, Mohammad & Chowdhury, Md Nurul. 2011. Design and Implementation of an Articulated Robotic Arm for Precise Positioning. Advanced Materials Research. Advanced Manufacturing Systems.
4. Peter Corke. 2011. Robotics, Vision and Control. Springer, Vol. 73.
5. Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. 2006. Robot Modeling and Control. JOHN WILEY, SONS, INC.
6. Hartenberg, Richard Scheunemann; Denavit, Jacques. 1965. Kinematic synthesis of linkages. McGraw-Hill series in mechanical engineering. McGraw-Hill. New York:
7. S. Mohamed, Ihab & Kumar, Ashwin & Serrano, Vaness. 2016. Serial Link Manipulator With 6-DoF.
8. ROBOTIS, e-Manual, Dynamixel. 2020. MX-28T/R/AT/AR. (https://emanual.robotis.com/docs/en/dxl/mx/mx-28/ , accessed 20/11/2020).
9. ROBOTIS, e-Manual. 2020. OpenCM 9.04 (https://emanual.robotis.com/docs/en/parts/controller/opencm904/ , accessed 20/11/2020).
10. ROBOTIS, e-Manual. 2020. OpenCM 485 EXP (https://emanual.robotis.com/docs/en/parts/controller/opencm485exp/ , accessed 20/11/2020)