Simulation and control of three degrees of freedom robot based on Comau smart5 NJ165

Sina Hasani Sadi

Department of electrical engineering

sina_hs@aut.ac.ir

Romina Alishah

Department of electrical engineering

elromina79@gmail.com

Shakila Kazempour Dizaji

Department of electrical engineering

shakila_kp@aut.ac.ir

Mahsa Beglari

Department of electrical engineering

mahsabeglari@aut.ac.ir

Parisa Fakhre Tabatabaie

Department of electrical engineering

Par_fr@aut.ac.ir

Mohamad Mahdi Malverdi

Department of electrical engineering

mahdi.ma78@aut.ac.ir

Mahdi Keshtani Mehrzad

Department of electrical engineering

m.mehrzad@aut.ac.ir

Abstract__ In this article, a three-degree-of-freedom robot is designed by SOLIDWORKS, then we attempted to implement the robot practically, and finally, controlled the robot using Arduino and MATLAB.

INTRODUCTION

Nowadays, industrial robots play an important role in performing complex operations at high speeds. This category of robots must have the ability to perform precise and sensitive operations continuously and repeatedly. However, robot manufacturers make controllers available to users so that they can improve their capabilities and computing power. On the other hand, the scientific community requires conducting numerous researches and improvements on systems in a short period of time. In this project, work will be done on the Comau Smart5 NJ165 robot.

To simplify the project in our practical work, we modeled a three-degree-of-freedom robot based on the first three degrees of freedom of Comau Smart5 NJ165 on a smaller scale.

Considering that the last three joints of their axes pass through the same point, it does not affect the movement of the end effector and only determines its orientation.

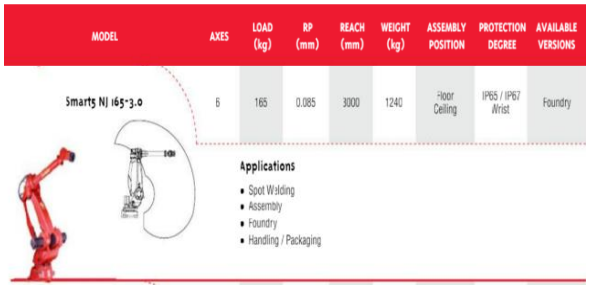

Figure 1: Comau Smart5 basic info

Figure 2: Comau Smart5 NJ165

The first step to implement the robot is to design it in software such as SOLIDWORKS, then links of the robot can be printed with help of 3D printing technology.

The main issue here was to keep the weight of the robot as low as possible but at the same time, it had to be large enough for motors to be placed on it. By using knowledge of the mechanics of materials we tried to lower the weight by omitting extra parts of the robot and making links hollow. Also, added some beams to the weak points to increase the strength of those points against loads and stresses acting on them.

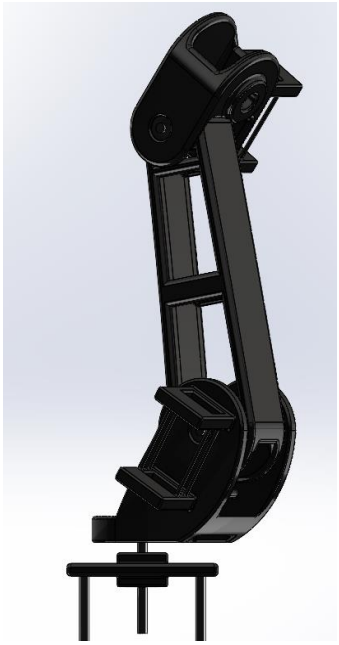

Figure 3: Simulation in SOLIDWORKS

I. Robot design

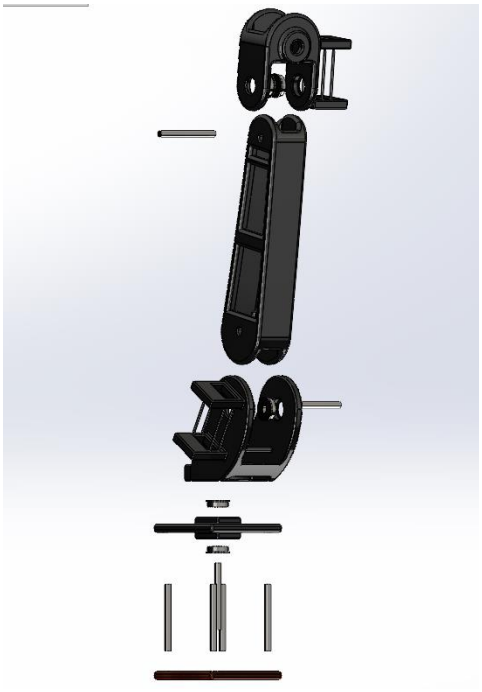

Figure 4: Robot links

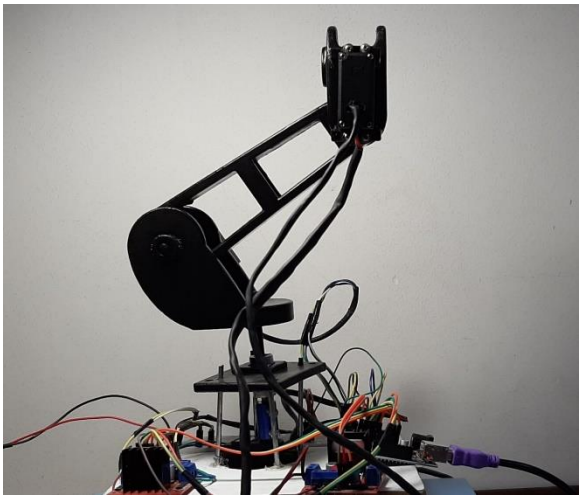

Figure 5: Assembled robot

II. Forward Kinematics

To control the arm, we needed forward and inverse kinematics of the robot. Results are shown below:

$$T_0^3 = \begin{bmatrix} \sigma_1 & \sigma_3\sin(\theta_3) - \sigma_2\cos(\theta_3) & 0 & -0.146875(\sin(\theta_2)\cos(\theta_1) + \sin(\theta_1)\cos(\theta_2)) - 0.03125(\sigma_3\cos(\theta_2) + \sigma_2\sin(\theta_2)) \\ \sigma_2\cos(\theta_3) - \sigma_3\sin(\theta_3) & \sigma_1 & 0 & -0.146875(\cos(\theta_2)\cos(\theta_1) + \sin(\theta_1)\sin(\theta_2)) - 0.03125(\sigma_2\cos(\theta_3) + \sigma_3\sin(\theta_3)) \\ 0 & 0 & 1 & 0.015 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\sigma_1 = -\sigma_3 sin(\theta_3) - \sigma_2\, cos(\theta_3)$$

$$\sigma_2 = cos(\theta_1)\, cos(\theta_2) - sin(\theta_1)\, sin(\theta_2)$$

$$\sigma_3 = cos(\theta_1)\, sin(\theta_2) - sin(\theta_1)\, cos(\theta_2)$$

End Effector Position:

$$P = \begin{bmatrix} -0.146875(\sin(\theta_2)\cos(\theta_1) + \sin(\theta_1)\cos(\theta_2)) - 0.03125(\sigma_3\cos(\theta_3) + \sigma_2\sin(\theta_3)) \\ -0.146875(\cos(\theta_2)\cos(\theta_1) + \sin(\theta_1)\sin(\theta_2)) - 0.03125(\sigma_2\cos(\theta_3) + \sigma_3\sin(\theta_3)) \\ 0.015 \end{bmatrix}$$

Transform matrix is obtained based on Classical Denavit Hartenberg method.

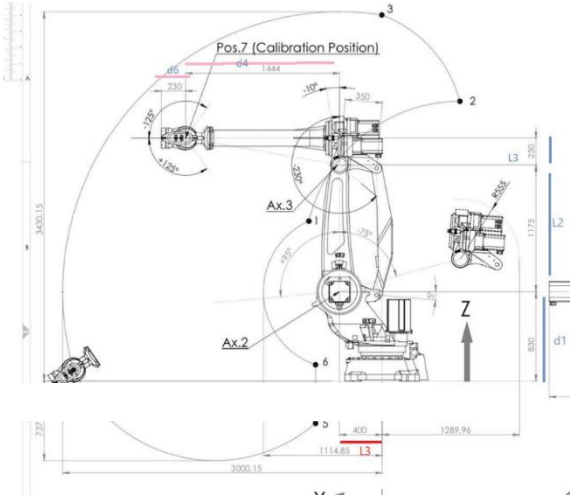

Figure 6

III. Inverse Kinematics

$$P_{ee} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\theta_1 = atan2(x, y)$$

$$\theta_3 = cos^{-1}\left(\frac{x^2 + y^2 + (z - 0.015)^2 - 0.0225488281}{0.0091796875}\right)$$

$$\theta_2 = tan^{-1}\left(\frac{0.0045898438\sin(\theta_3) + 0.0004882813\sin(2\theta_3) \pm (z - 0.015)\sqrt{x^2+y^2}}{x^2+y^2+0.0215722656+0.0009765625cos^2(\theta_3)+0.0091796875\cos(\theta_3)}\right)$$

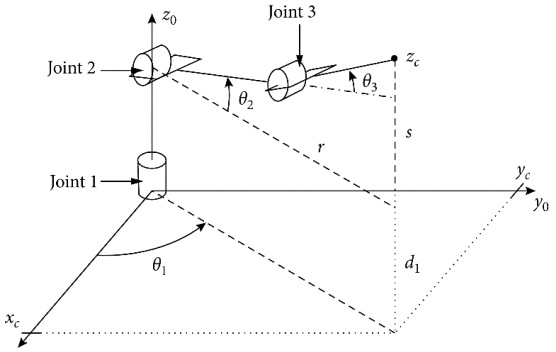

Figure 7

IV. Mass and Inertia Matrix

The mass of each link and its matrix of inertia around its center of mass is also calculated for further use:

Mass (kg):

Link 1: $m_1 = 0.0375$

Link 2: $m_2 = 0.10235$

Link 3: $m_3 = 0.0548$

Moment of Inertia (kg.m^2):

Link 1: $I_1 =$
$$\begin{bmatrix} 0.0009 & -0.00003 & 0.00001 \\ -0.00003 & 0.0001 & -0.00001 \\ 0.00001 & -0.00001 & 0.00009 \end{bmatrix}$$

Link 2: $I_2 =$
$$\begin{bmatrix} 0.0004 & 0 & 0 \\ 0 & 0.00033 & 0 \\ 0 & 0 & 0.00034 \end{bmatrix}$$

Link 3: $I_3 =$
$$\begin{bmatrix} 0.00004 & 0 & 0 \\ 0 & 0.00003 & 0 \\ 0 & 0 & 0.00005 \end{bmatrix}$$

V. Robot programming with inverse kinematics

The next step was to control the arm. For this manner, we have used three servo motors. We removed the controller board of each motor and connected the motors to L298N motor drivers to provide enough voltage for running the motors. We connected the drivers to a 12-volt dc source and pins of Arduino Uno board. To program the board, we used the Simulink environment of MATLAB and the Simulink support package for arduino hardware. The block diagram designed is shown below:

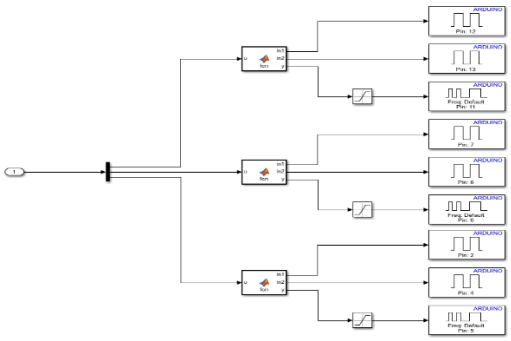

Figure 8: PWM block diagram

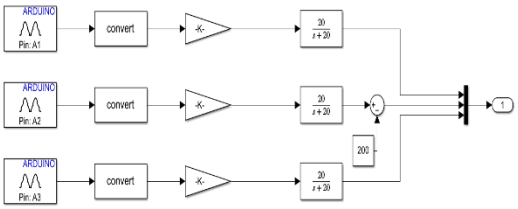

Figure 9: Feedback block diagram

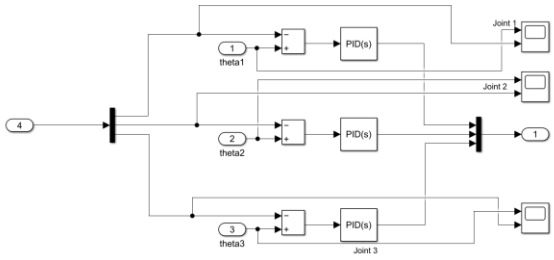

Figure 10: Controller block diagram

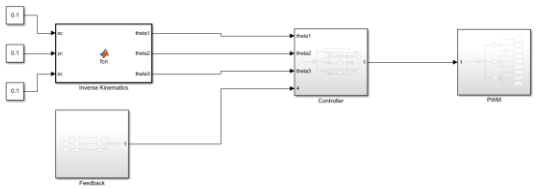

Figure 11

The idea behind the Simulink model we made is that the desired point that the end effector wants to reach is given to the model in the Cartesian space. But we can only control the angles of the motors. So, we used the equations which were derived at the inverse kinematic part to obtain the desired angle of each link from desired end effector position. This is what the inverse kinematics block does. It actually gets the ee position and gives the desired angles as its output.

These angles go to the controller block as input signals to be compared with the feedback signals received from the potentiometers to get the error and generate the necessary output signal to be sent to the PWM block by PID control.

In the feedback block, the data coming from the potentiometer is processed and sent to the controller as feedback.

Also, calibration of angles of motors relative to joint angles is done in this block by summing constants with the feedback signals.

In the PWM block, using the output signals of the controller, the required direction and rotation speed of each motor are calculated and applied to the motor to reach the desired point.

VI. Conclusion

There were many challenges in the way of implementing the robot for example, the Arduino board did not apply the necessary voltage to start the motors ,this problem was solved by a motor driver and a 12 volt power supply .

Another problem was related to costs, the dimensions of robot should be appropriate to reduce the weight of the robot .

The diameter of the shaft in engine is considered 5 mm , therefore , one side of the couplings should be 5 mm and the other side should be 6 mm and this type of coupling is rare , we drilled 5mm couplings until one side becomes 6mm.

REFERENCES

[1] John J. Craig, Introduction to Robotics Mechanics and Control, Third Edition

[2] Mark W. Spong, Robot Modeling and Control