Hindawi
Journal of Robotics
Volume 2017, Article ID 7508787, 11 pages
https://doi.org/10.1155/2017/7508787

Research Article

Generation and Control of Basic Geometric Trajectories for a Robot Manipulator Using CompactRIO®

Jorge Luis Aroca Trujillo,[1] Alexander Pérez-Ruiz,[2] and Ruthber Rodriguez Serrezuela[1]

[1] *University Corporation of Huila, Corhuila, Neiva, Colombia*
[2] *Escuela Colombiana de Ingeniería Julio Garavito, Bogotá D.C., Colombia*

Correspondence should be addressed to Ruthber Rodriguez Serrezuela; ruthber.rodriguez@corhuila.edu.co

Received 16 September 2017; Accepted 9 November 2017; Published 11 December 2017

Academic Editor: L. Fortuna

The utility of a robot manipulator focuses on the ability to locate its end effector in a position with a determined orientation following a specified trajectory. For this, algorithms were used in order to generate and control the movements joints of robot in a synchronized way. The high-level languages to program robots are based on three types of movement: joint interpolation (MOVEJ), linear interpolation (MOVES), and circular arcs (MOVEC), which are used to develop any type of task. In this work, these three movements are implemented in the industrial controller CompactRIO, as part of the reconditioning process of a robot manipulator of five degrees of freedom (5 DOF) whose controller was obsolete. As a result, it will have an interface in LabVIEW where you can view and modify the basic parameters implemented in the industrial controller. In addition, the results of the validation tests of the joint positions and the end effector of the manipulator will be found.

1. Introduction

The robot manipulators have been constituted as an essential part of modern industries. Some of the reasons are the low production cost, the optimization in the processes, their flexibility, and the adaptation to several productive situations. Additionally, the robot manipulator with only small modification in the programming allows industries to adopt this technology in their operations instead of company staff. Generally, the algorithms designed for the monitoring and control of trajectories must be accompanied by the mathematical support that guarantees the good performance of the scheduled tasks. There are techniques with different approaches to control robot manipulator movements, such as neural networks [1], uncoupled control [2], and game theory [3]. However, some control units that currently accompany these manipulators and incorporate these techniques, due to the passage of time, have presented faults or are very outdated, which limits their use.

In this article, we intend to endow a robot manipulator of five degrees of freedom (5 DOF) a new industrial controller (CompactRIO) (Figure 1), which allows designing a modern interface with open control algorithms. This new control unit offers the possibility of using a generic or different industrial controller to factory design for robot manipulators, which are functional, reliable, and user friendly.

In this new control unit, three types of path-generating algorithms are implemented, which provide the positions that a robot manipulator must follow to execute a specific task [4, 5]. The first algorithm develops a joint path (MOVEJ), in which the end effector does not have any relevance. The other two will generate Cartesian trajectories that will draw with the end effector the shape of a line or an arc (MOVES and MOVEC). The programming will be done through LabVIEW and will be implemented in the same control unit. It will interact with the user through a computer connected remotely to the Ethernet port, which will make it easier to visualize, create, and modify the tasks that the user would like the manipulator to replicate. The developed interface will contain the reading of the commands, the mathematical modeling found, the algorithms for generating trajectories, the visualization of variables, and the position control (PID) for motors [6].

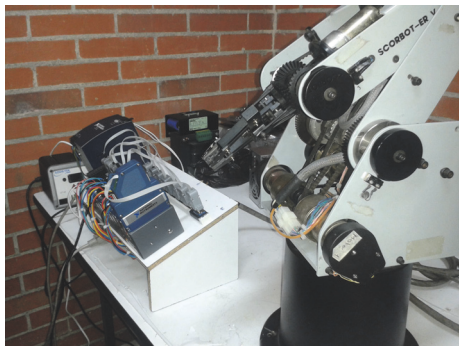


Figure 1: Robot manipulator Scorbot ER_4PC.

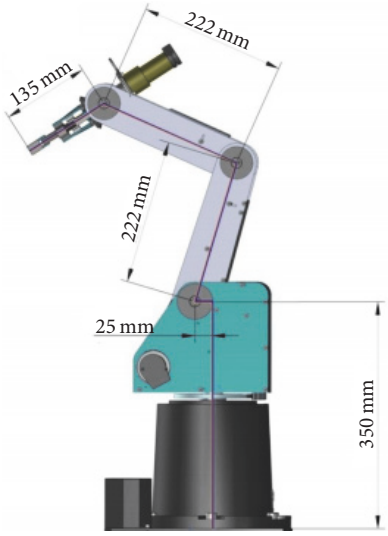


Figure 2: Mechanical structure of the Scorbot manipulator [7].

Figure 3: Industrial controller, NI CompactRio [retrieved on June 8, 2017, from http://www.ni.com/compactrio/value-controller/esa/].

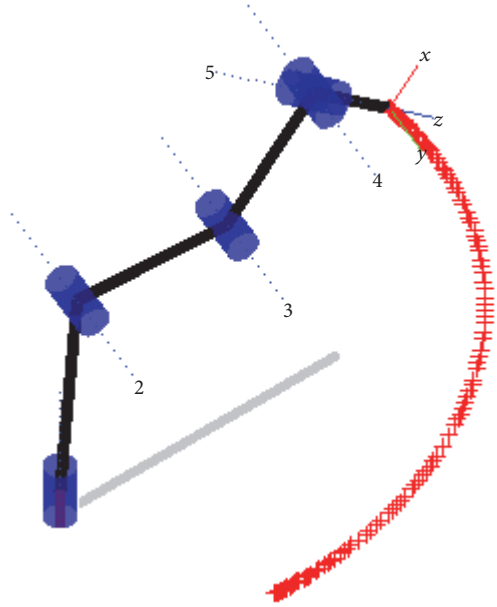


Figure 4: Simulation of movement joint.

2. Hardware Description

2.1. Robot Manipulator. The Scorbot manipulator (Figure 2) has an anthropomorphic configuration, composed of a mechanical structure with five joints, DC motors of 12 Volts, rotary encoders coupled to the axes of the motors, and five microswitches to determine the position of home.

The anthropomorphic design provides a broad region of end effector operation [8, 9]. It has a rotation at the base of 310°, shoulder rotation of +130°/−35°, rotation of the elbow of ±130°, tilting of the clamp of ±130°, and rotation of the mechanically unlimited clamp but originally restricted to ±570° electrically (configurable software) [5].

The Scorbot-ER 4PC has several types of transmission to generate the movement in the joints of the robot; these transmissions are composed of gears with different configurations of trains, pulley, and toothed belts. It also has a spindle transmission in charge of opening and closing the clamp.

2.2. Industrial Controller: CompactRio. Manufactured by the company *National Instruments*, it provides reliable and deterministic performance for real-time monitoring and control applications. It has a robust and modular architecture, with a reconfigurable FPGA chassis capable of executing high-speed processes, which is interconnected with the LabVIEW (*Laboratory Virtual Instrumentation Engineering Workbench*) development platform (Figure 3) [10].

3. The Algorithms of Trajectories Implemented Description

Three movements (MOVEJ, MOVES, and MOVEC) will be dealt with, which will be implemented in the industrial controller CompactRio, which can be combined so that the robot can develop a specific task [11].

3.1. Joint Movement: MOVEJ. The use of this movement allows the manipulator to reach an end position, with independent joint movements. This does not ensure a linear trajectory between the initial and final points (Figure 4); that is to say the end effector moves from one point to another by a random path [12].

The coordinates in the end effector workspace can be obtained through the Direct Kinematic Model (DKM) [13–15], using the joint values executed in the MOVEJ command. The pseudo-code can be seen in Algorithm 1.

Data: Actual position: $[\theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5]$,
desired position: $[\theta_{1d} \ \ \theta_{2d} \ \ \theta_{3d} \ \ \theta_{4d} \ \ \theta_{5d}]$
Result: New position $q = [\theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5]$
(1) **while** $[\theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5] \neq [\theta_{1d} \ \ \theta_{2d} \ \ \theta_{3d} \ \ \theta_{4d} \ \ \theta_{5d}]$ **do**
(2) **if** $\theta_1 \neq \theta_{1d}$ **then**
(3) $\theta_1 = \theta_1 \pm \Delta\theta_1$;
(4) **end**
(5) **if** $\theta_2 \neq \theta_{2d}$ **then**
(6) $\theta_2 = \theta_2 \pm \Delta\theta_2$;
(7) **end**
(8) **if** $\theta_3 \neq \theta_{3d}$ **then**
(9) $\theta_3 = \theta_3 \pm \Delta\theta_3$;
(10) **end**
(11) **if** $\theta_4 \neq \theta_{4d}$ **then**
(12) $\theta_4 = \theta_4 \pm \Delta\theta_4$;
(13) **end**
(14) **if** $\theta_5 \neq \theta_{5d}$ **then**
(15) $\theta_5 = \theta_5 \pm \Delta\theta_5$;
(16) **end**
(17) $q = [q; \ \theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5]$;
(18) **end**
(19) New position $q = [\theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5]$;

Algorithm 1: Algorithm for joint movement of MOVEJ.

3.2. Linear Motion: MOVES. If it is desired to arrive at a Cartesian position following a trajectory in a straight line with the end effector, the command MOVES is used. This command requires a vector with six values; the first three are Cartesian variables (X, Y, and Z) of the desired position in the workspace of the origin of the coordinate system of the end effector. The following three refers to your orientation; this orientation is entered with the angles Roll(δ), Pitch(β), and Yaw(γ), also known as angles RPY [16].

In this way, you will have a linear movement as Figure 5 [12], through the path generated from the current point or from start to end point (point entered with the command).

For this command, we will briefly discuss the two transformations that must be performed [17]. The treatment is done differently, mainly because the space of translations is Euclidean, while the space of rotations is not.

Comparing the two movements, translation is the simplest of the two interpolations, since it is expected that a point $P_o \in \mathbb{R}^3$, denoted by three variables (x_o, y_o, and z_o) defined in the Cartesian space, is moved through a translation vector (Δx, Δy, and Δz).

The interpolation of orientation and rotation is done by the mathematical method SLERP (*Spherical Linear Interpolation*) (it is a unitary sphere of $\mathbb{R}^4$) [17–19]. For this, the orientation and rotation must first be represented in four dimensions or quaternions and then interpolated by the method SLERP. This method is to perform the movement at constant speed along a circle, on the surface of a hypersphere (Figure 6) [10], under the assumption that the positions of the trajectories of the end effector are given between the points of P_o and P_f. The effect is a uniform rotation with angular velocity around a fixed axis of rotation, which guarantees

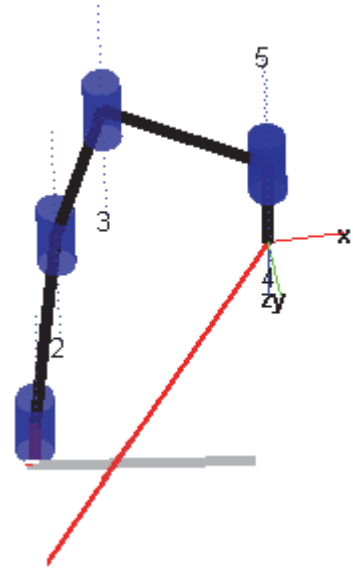


Figure 5: Simulation of motion in a straight line.

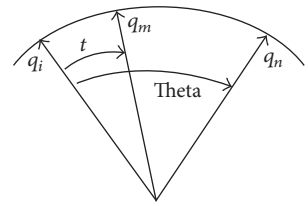


Figure 6: Spherical linear interpolation.

Data: Transformations Matrix T_o, T_f
Result: Interpolated positions $[\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5]$
(1) $p = 0 : 1 : (|P_f| - |P_o|)$;
(2) $N = \text{length}(p)$;
(3) $t = 0 : (0.1\,(N/(N-1))) : N$;
(4) $P_f = T_f(1:3, 4)$;
(5) $P_o = T_o(1:3, 4)$;
(6) $R_f = T_f(1:3, 1:3)$;
(7) $R_o = T_o(1:3, 1:3)$;
(8) **for** $i = 1 : 1 : N$ **do**
(9) $\quad P_n = P_o + (P_f - P_o) \cdot t(i)$;
(10) $\quad q_o \leftarrow R_o$;
(11) $\quad q_1 \leftarrow R_1$;
(12) $\quad \theta = \arccos(q_o \cdot q_1)$;
(13) $\quad q_m = q_o \cdot \sin((1 - t(i)) \cdot \Omega)/\sin(\Omega) + q_1 \cdot \sin(t(i) \cdot \Omega)/\sin(\Omega)$;
(14) $\quad R_n \leftarrow q_m$;
(15) $\quad T_n = [R_n \ \ P_n; \ 0 \ 0 \ 0 \ 1]$;
(16) $\quad q = [\text{IKM}(T_n)]$ ⇝ Inverse Kinematic Model
(17) **end**
(18) Interpolated positions $q = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5]$;

Algorithm 2: Algorithm for joint movement of MOVES.

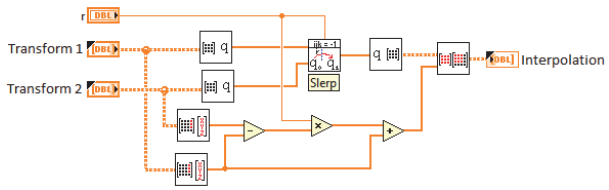


Figure 7: Interpolation implemented in *LabVIEW*.

a single route [18, 20]. Equation (1) [21] represents such movement.

$$q_m = \frac{q_o \cdot \sin((1 - t(i)) \cdot \Theta) + q_1 \cdot \sin(t(i) \cdot \Theta)}{\sin(\Theta)}, \quad (1)$$

where q_m is *quaternion* interpolated, q_o is first quaternion, q_1 is second quaternion, Θ is the angle between the vectors q_o and q_1, and t is the spherical interpolation between vectors q_o and q_1, with values between 0 and 1.

With this method, implemented in Algorithm 2, any interpolation of quaternions can be drawn to make a smooth movement of the rotation of an object with complex rotations and the orientation of the same ones. Once the interpolated matrices are obtained, we proceed to change Cartesian coordinates to articular coordinates using the Inverse Kinematic Model (IKM) [22, 23].

Algorithm 2 shows the description of the execution of the MOVES command whose main part of the implementation can be seen in Figure 7.

3.3. MOVEC: Circular Movement. Three linearly independent points are indispensable to do this movement. The first is the value of X, Y, and Z of the Cartesian plane where the end effector is located, the second is the destination point, and

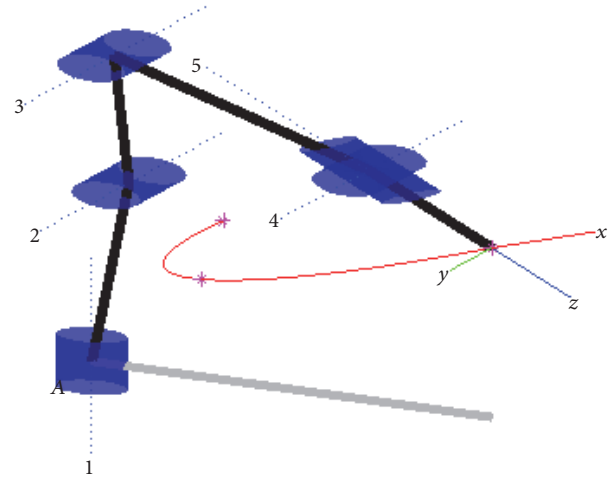


Figure 8: Arc and circular movement.

the third belongs to the via point. That last one, in addition to helping in defining the path where the end effector should slide, defines the plane where it is going to work, as shown in Figure 8.

To perform the stroke, we define the orientation and direction of a vector associated with the systems X_1, Y_1, and Z_1. The representation is made by a vector in a frame that rotates angle θ around X, an angle α around Y, and an angle β around Z and moves the distances **a**, **b**, and **c** with respect to the axes x, y, and z; all this transformed to the coordinates of the frame of reference. On the frames X_1, Y_1, and Z_1, the three linearly independent points define the plane that will contain the distribution of the points that make up the curve.

Data: P_f, P_v
Result: Interpolated positions $[\theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5]$
(1) Vectors a and $b \mapsto a = (P_f - P_o) \leftrightarrow b = (P_v - P_o)$;
(2) $x = a/\text{norm}(a) \mapsto$ Value in x associated to the system to O_1;
(3) $z = \text{cross}(a, b)$;
(4) $z = z/\text{norm}(z) \mapsto$ Vector normal to the plane or z associated to the system to O_1;
(5) $y = \text{cross}(z, x) \mapsto$ Value in y associated with the system to O_1;
(6) $\Theta = \text{acos}(\text{dot}(a, b)/(\text{norm}(a) * \text{norm}(b))) \mapsto$ Angle between vectors a and b;
(7) $x_{vp} = \text{norm}(b) * \cos(\Theta)$;
(8) $y_{vp} = \text{norm}(b) * \sin(\Theta)$;
(9) $X_{fp} = \text{norm}(a)$;
(10) $m = -x_{vp}/y_{vp}$;
(11) $x_{cp} = P_{fp}/2 \mapsto x$ of the center of the circumference;
(12) $y_{cp} = (y_{vp}/2) + m(x_{cp} - (x_{vp}/2)) \mapsto y$ of the center of the circumference;
(13) $r = \text{norm}([X_{fp} \ \ 0] - [x_{cp} \ \ y_{cp}])$;
(14) $\alpha_1 = -\text{atan2}(y_{cp}, x_{cp}) \Rightarrow \alpha_2 = \pi - \alpha_1$;
(15) $\sigma = \text{linspace}(\alpha_2, \alpha_1, N)$;
(16) $y_s = r * \sin(\sigma) + y_{cp} \mapsto x$ interpolated from the circumference;
(17) $x_s = r * \cos(\sigma) + x_{cp} \mapsto y$ interpolated from the circumference;
(18) $T = [x \ \ y \ \ z \ \ P_o; \ \ 0 \ \ 0 \ \ 0 \ \ 1]$;
(19) **for** $i = 1 : N$ **do**
(20) $P = T * [x_s(i) \ \ y_s(i) \ \ 0 \ \ 1]'$;
(21) $c = P(1:3) - P_o$;
(22) $x_n = c/\text{norm}(c)$;
(23) $y_n = \text{cross}(z, x_n)$;
(24) $Tc = [x_n \ \ y_n \ \ z \ \ P(1:3); \ \ 0 \ \ 0 \ \ 0 \ \ 1]$;
(25) $q = [\text{IKM}(Tc)] \rightsquigarrow$ Inverse Kinematic Model
(26) **end**
(27) Interpolated positions $q = [\theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5]$;

Algorithm 3: Algorithm for joint movement of MOVEC.

The circumference in the plane is drawn from the perpendicular bisector of P_oP_v and the perpendicular bisector of P_oP_f, where they will be cut into a single point, which is the center of the circumference, passing through P_o, P_v, and P_f, since the three are equidistant from it (Figure 9). From another point of view, it may be said that it is necessary to find a circumference circumscribed in the triangle formed by the three points, the center of this circumference is obtained through the intersection of the two-side bisectors of the triangle. The problem generates as a result a circumference that tends to infinity.

Similarly if you wish to complete the drawn arc to form a complete circle, it is necessary to generate another movement on the same plane, which has, as its starting point, the end point of the first and the end point, as the starting point of the first command. The diameter is defined by the distance between the start and end points and in turn must be equal to the distance between the two-way points. If this proportion is not met, it is also possible to obtain an ellipsoidal trace whose distance between its height and width can vary depending on the distances of the initial and final points with respect to the distances of the way points (or vice versa). The operations that they use to implement the MOVEC command are defined in Algorithm 3.

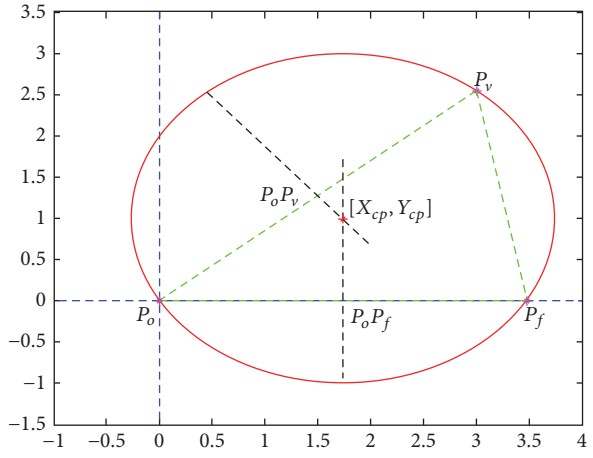


Figure 9: Construction of the circle, using the perpendicular bisectors P_oP_v and P_oP_f.

4. Scheme Implemented

The implementation of the algorithms of trajectories for the robot manipulator required the implementation of the

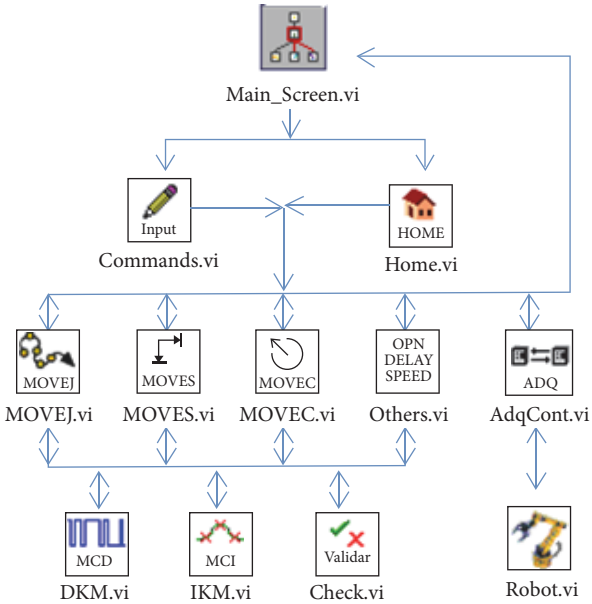


Figure 10: SubVI scheme for robot arm control.

graphical interface that incorporates the acquisition and control of variables supplied by the industrial controller, the input of parameters for the generation of the movements, and the algorithms generators of trajectories, among others. Next, a brief description will be given on the most relevant components that are part of the work done.

4.1. General Scheme of the Interface Implemented. To integrate the mechanical system of the robot and the industrial controller CompactRIO, to the environment of LabVIEW, the programming and interconnection scheme of the figure was developed in Figure 10, which is programmed in the industrial controller [24].

The main VI of the system is completely designed and intended to interact with the user. To do this, it receives and displays the articulator positions of the manipulator, together with the Cartesian positions of the end effector. In addition, it sends the commands that are to be executed to the two blocks responsible for interpreting them (Commands.vi and Home.vi).

The VI Home is just responsible for providing the articular positions to bring the manipulator to the initial configuration (Home). On the other hand, the VI Commands organizes and sends the other orders to the VIs in charge of executing each of the algorithms (MOVEJ, MOVES, and MOVEC), the desired speed with which to execute each robot movement (SPEED), the time delay between commands (DELAY), and the opening and closing of the fingers of the clamp (OPEN and CLOSE).

As the names indicate, the MOVEJ, MOVES, and MOVEC VIs, with the help of the DKM, IKM, and validate blocks, are in charge of creating the algorithms generating articular, linear, and circular trajectories, respectively. These three first blocks give the point to point the route that must be made to comply with the command, the acquisition block,

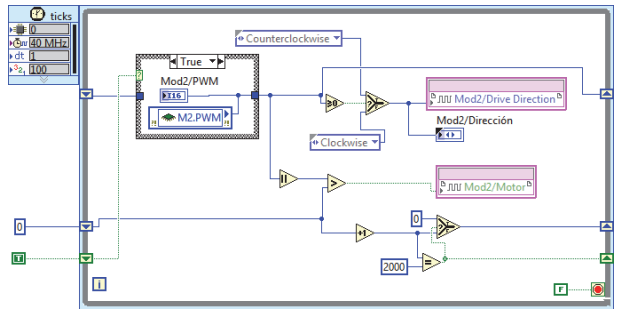


Figure 11: PWM signal control block.

and the control of variables (Adpcont.vi). The other VI is destined to interpret the commands SPEED, DELAY, OPEN, and CLOSE to later execute them in the manipulator.

4.2. Robot, CompactRio, and Computer Communication. NI 9503 FPGA modules will be responsible for the acquisition of the signals generated by the encoders, through the entrances: phase A+ and phase B+. In addition, it will be responsible for energizing the motors that move each joint, through the PWM that each module counts. The signals from the switches, used to calibrate the positions, are read in the inputs 1, 2, 3, 4, and 5 of the NI 9403 digital input module.

These six modules, five to control the motors and one for the calibration switches, are connected to the industrial controller CompactRio, which, in turn, communicates with the computer through a network connection that owns the chassis.

4.2.1. Acquisition and Control of Signals. The purpose of this VI is to provide the program with variable reading and writing, such as the direction of rotation of the motors, number of pulses generated by the *encoders* increment, currents delivered to each motor, PWMs, stop, reset, and the activation of the push-buttons. The latter is obtained through the configuration of the NI-9403 module and the other variables are done through the NI-9505 modules.

To obtain the PWM signal with which the speed of each of the motors will be controlled, a block is created in a time frame, which is executed at a clock frequency of 40 Mhz (internal frequency of the CompactRio), as the picture shows in Figure 11.

In addition, the NI-9505 module delivers signals A and B of the encoders that are 90 degrees out of phase, so that the diagram in Figure 12 performs the counting of the pulses. Here, the spin direction is detected, based on the relationship phase between the signals, and then add or subtract them, according to the case.

With this count, a small operation is carried out to find the angle that the motor has turned.

$$\theta_{\text{motor}} = \frac{2\pi}{\text{Resolution\_of the\_encoder}} \#\text{pulses}. \quad (2)$$

The way the manipulator is constructed, with θ_3 and θ_4, presents an angle different from that measured on the axis of the motors that move them. In other words, these angles

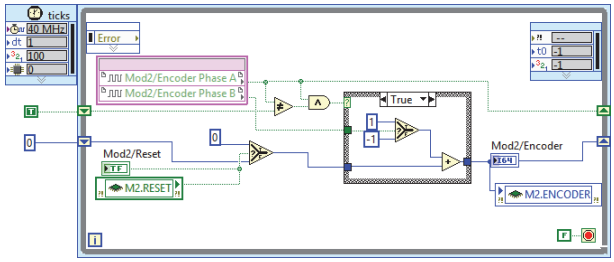


Figure 12: Structure used to read signals from encoders.

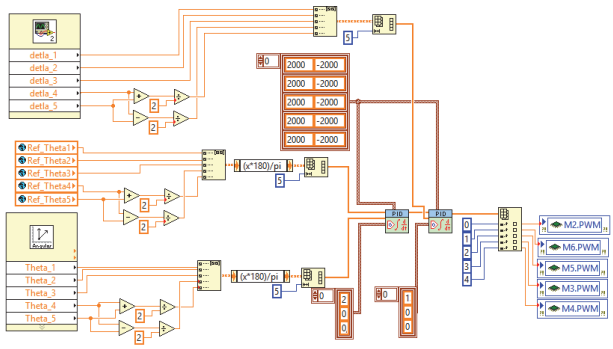


Figure 13: Decoupled control loop diagram.

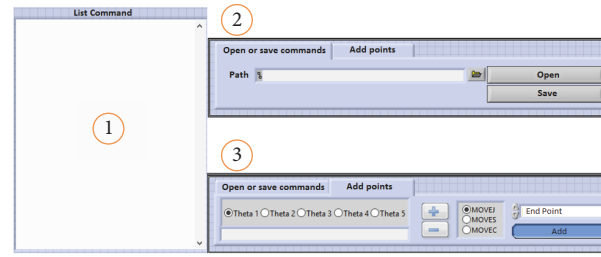


Figure 14: Enter orders to execute.

Table 1: Commands allowed.

Command	Operation
HOME	Press button
DELAY	DELAY $[t]$
MOVEJ	MOVEJ $[\theta_{1d} \ \theta_{2d} \ \theta_{3d} \ \theta_{4d} \ \theta_{5d}]$
MOVES	MOVES $[P_{xd} \ P_{yd} \ P_{zd} \ \delta_d \ \beta_d \ \gamma_d]$
MOVEC	MOVEC $[P_{xf} \ P_{yf} \ P_{zf} \ P_{xv} \ P_{yv} \ P_{zv}]$
SPEED	SPEED [%]
OPEN	OPEN
CLOSE	CLOSE

are affected by the previous angles; this indicates that, besides taking into account the angles of the motors, the angles that precede the one that needs to be found must be added or subtracted, in such a way that

$$\begin{aligned} \theta_3 &= \theta_2 + \theta_{\text{motor3}}, \\ \theta_4 &= \theta_2 + \theta_3 + \theta_{\text{motor4}} + \theta_{\text{motor5}}, \\ \theta_5 &= \theta_{\text{motor4}} - \theta_{\text{motor5}}, \end{aligned} \tag{3}$$

where $\theta_{\text{motor4}} + \theta_{\text{motor5}}$ generate the wrist movement that moves the end effector up and down. Additionally, the difference of these angles $(\theta_{\text{motor4}} - \theta_{\text{motor5}})$ produces the rotary movement of the tool; the latter is not affected by other angles, as is the case of θ_3 and θ_4.

The implemented control scheme was carried out through a simple control, which is based on the movement of the joints independently or decoupled (Figure 13), which together with the path-generating algorithms coordinates the movements articular for Cartesian trajectories.

The first control block is in charge of following the reference position given by the algorithms generators of trajectories; the second control block allows reaching the changes of speed that must make the manipulator guarantee the correct articular and Cartesian positions in different instants of time.

The PID controllers are limited to generate a maximum and minimum PWM of ±2000, which corresponds to the input of the DC motor at ±12 volts.

4.2.2. Enter Commands. The development of this VI allows entering the commands for the robot manipulator to perform a task. There are three options: the first is to write directly the command to be used in the command console of the main screen of the program; the second way is to import to the command console the flat file where you have the commands to execute; the last one requires moving the manipulator from grade to grade at each joint until the desired position is reached, then adding the point with the desired operation to the command console (Figure 14).

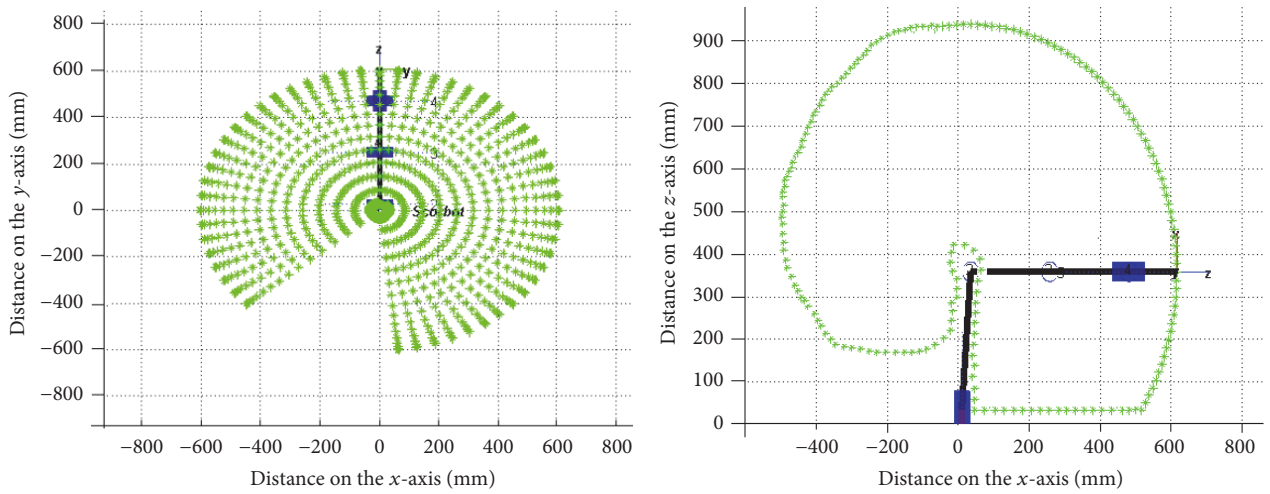
For correct use of the manual mode (option three), it is necessary to select the joint you want to move and then increase or decrease the angle with the plus (+) or minus (−) buttons. Once the robot is in the desired position, you can choose the operation to perform pressing the add button, which is added to the list. For MOVEJ and MOVES commands, only the end point needs to be stored, but for MOVEC it is necessary to store two points; one will be the end point and the other will be the way point.

The commands allowed and the correct way to enter them are mentioned in Table 1.

The angles θ_{1d}, θ_{2d}, θ_{3d}, θ_{4d}, θ_{5d}, δ_d, β_d, and γ_d are expressed in degrees, even though the execution of the program is done with radians, and the coordinates P are given in millimeters.

4.2.3. HOME Button. This command has the function of calibrating the angles of each joint and bringing the robot manipulator to an initial or reference position (home).

4.2.4. SPEED Command. This order provides the speed change with which the robot manipulator performs the movements, and the SPPED$[x]$ command is entered, where x is the integer value between 0 and 100 given in percent, where 100% supplies the maximum joint velocity of 0.218 [rad/s].



(a) Border of workspace outside area, top view

(b) Border of workspace, side view

Figure 15: Scorbot workspace found in simulation.

4.2.5. DELAY Command. This command forces the robot to remain in its last position during the seconds that it is commanded. For that, the block time delay of LabVIEW is used.

4.2.6. OPEN and CLOSE Commands. With these commands, the manipulator can open and close the clamp, which makes it possible to hold and release objects.

4.2.7. MOVEJ, MOVES, and MOVEC Commands. These commands have the function of generating the interpolations so that the manipulator can trace desired trajectories, as previously exposed.

5. Obtained Results

5.1. Workspace. The effective working space is directly related to the length of the links, the mechanical limitations, and the configuration that the robot has. It can be found mathematically, through the equations describing the position of the end effector along with the mechanical limitations it possesses [25].

Physically, the workspace indicates that it is able to reach these points without exceeding the joint limits. This is essential to be able to propose the movements that are required to develop a task.

Due to the configuration that the Scorbot manipulator has (anthropomorphic), the movement described by the end effector is realized in the three-dimensional space.

The graphs corresponding to the workspace of the robot manipulator Scorbot-ER, with the new restrictions implemented by software in this work, are shown in Figures 15(a) and 15(b). In such graphs, some of the successions of points of the Cartesian coordinates found from the articular positions of the robot are presented. In this way, it gives an idea of the points, in the Cartesian system, that can be reached. For the manipulator to be controlled, most of the positions can be got that are within a radius less than or equal to 604 mm.

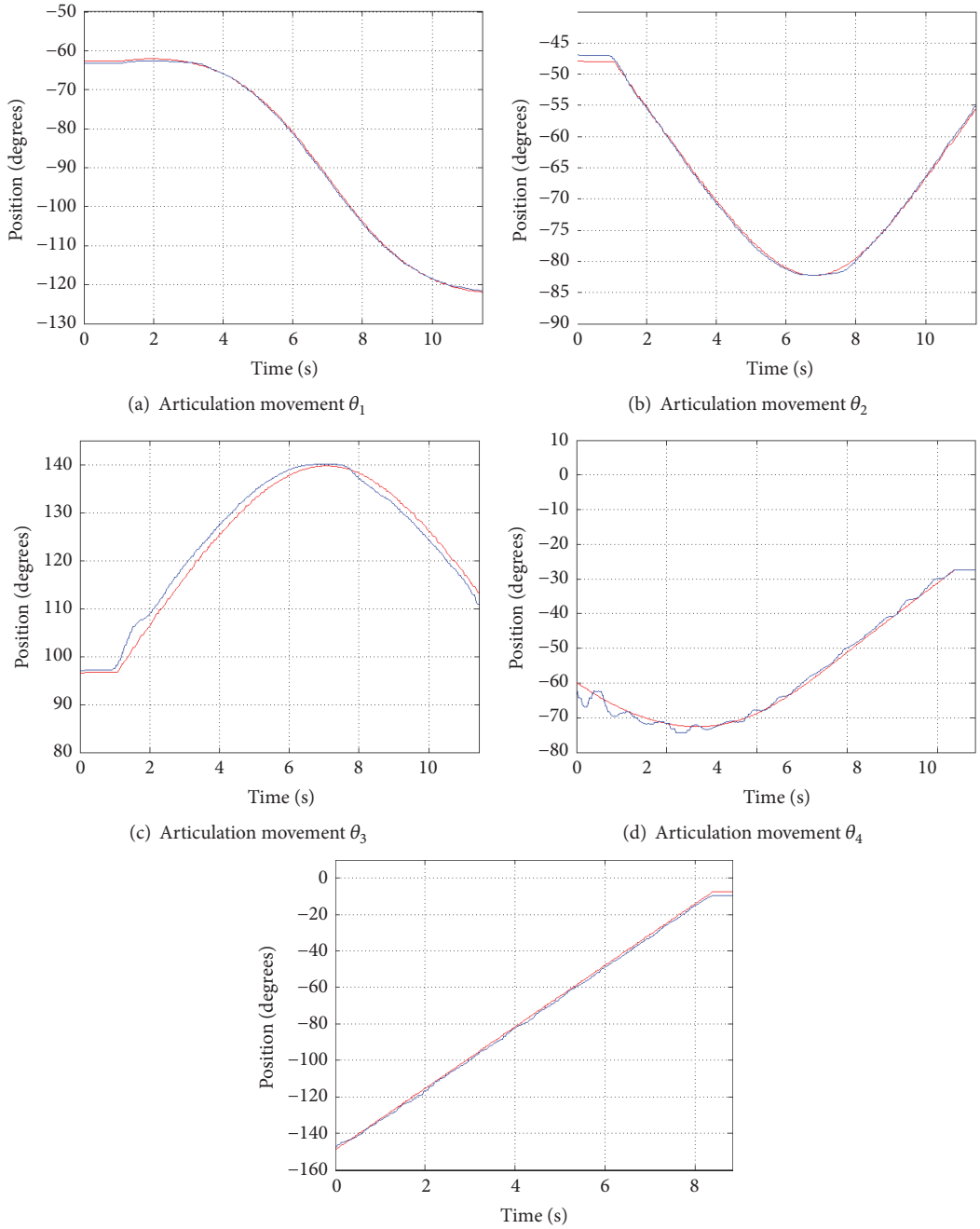
5.2. Performance of Joint Movement. An analysis of the result of the kinematic model is performed, based on the MOVEJ command. For that, a known position and a desired joint position are used. For the test, the starting point is chosen as $\theta_0 = [0, 0, 0, -90, 0]$, and the desired joint position is designated, $\theta_d = [90, -10, -45, 0, 0]$, as the point of arrival or end, where angles are given in degrees. With these values, an interpolation of the initial angles to the desired angles is generated (Figure 4).

Afterwards, a comparison of the data of the movement made by the manipulator and those obtained in the simulation is made. Figures 16(a), 16(b), 16(c), 16(d), and 16(e) perform such joint movements generated by the robot, where the red stroke represents the expected values and the blue stroke the articular path made by the manipulator.

5.3. Performance of Linear Motion in the Cartesian Space. A straight stroke is the result of applying the MOVES command (Figure 5), as detailed during the development of this document. The example used to analyze the performance of the movement is performed with the initial coordinates, $[-28.6, -351.04, 64.24, 3.13, 0, -1]$, to the desired point of $[280, -4, 305, 90, 0, -6.3]$, where distances are given in millimeters and angles in degrees.

The desired movement for the end effector is seen in Figures 17(a), 17(b), and 17(c), which describes the path taken in the coordinate axes x, y and z, where the red stroke represents the expected stroke and the blue the actual behavior of the manipulator.

5.4. Cartesian Movement Performance: Circular. Like the previous move, in this command, a planned figure is described



(a) Articulation movement θ_1

(b) Articulation movement θ_2

(c) Articulation movement θ_3

(d) Articulation movement θ_4

(e) Articulation movement θ_5

Figure 16: Joint movement of the MOVEJ command.

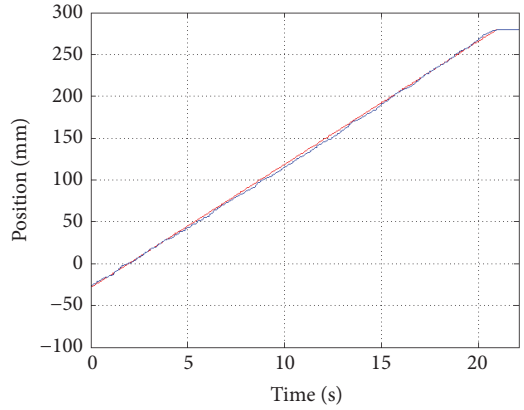
on the Cartesian plane. This movement realizes a circular stroke realized by the end effector during its trajectory. The starting point was Cartesian coordinates $X_o = [145, -276, 185]$, an obligatory point of passage (way point) $X_v = [25, -170, 222]$, and a point of arrival $X_f = [-140, -235, 190]$.

Consequently, the movements of the coordinate axes shown in Figure 18 present some errors of position of the end effector, due to the response time that has the manipulator, comparing it with the simulation (red line).

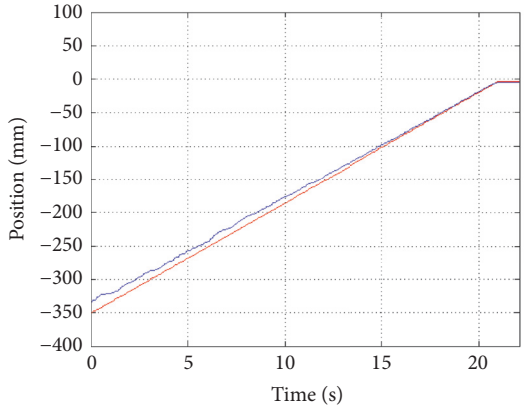
6. Conclusion

Having programmed the path generator algorithms and position control system in the CompactRio controller allows the manipulator to continue the task it is developing when there is a connection failure between the computer and the controller.

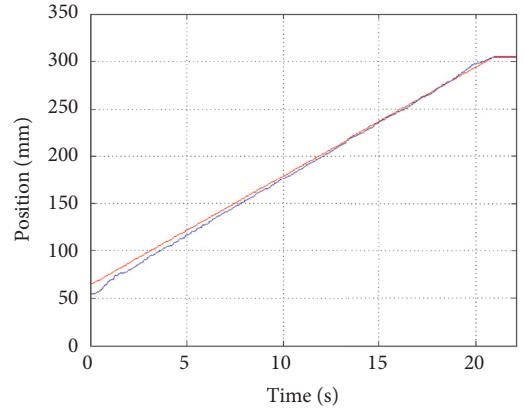
Three path algorithms are implemented, MOVEJ, MOVES, and MOVEC, which provide a sequence of points in the joint space to generate the different movements



(a) Shaft movement X

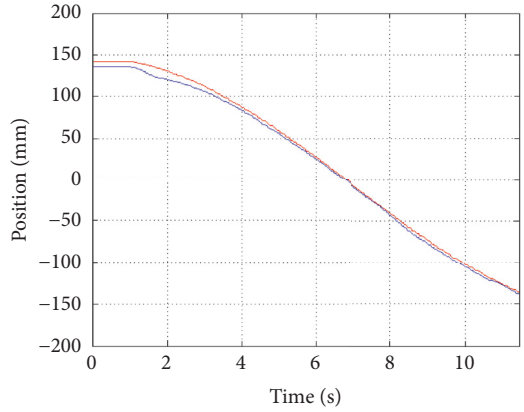


(b) Shaft movement Y

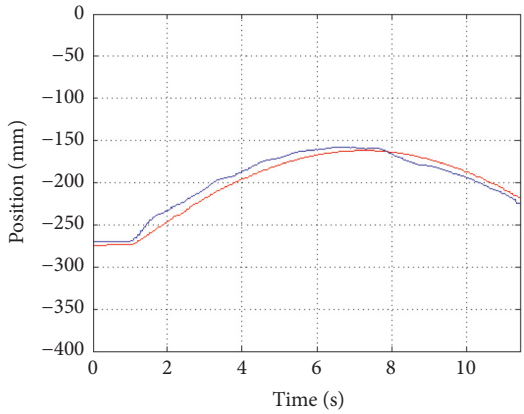


(c) Shaft movement Z

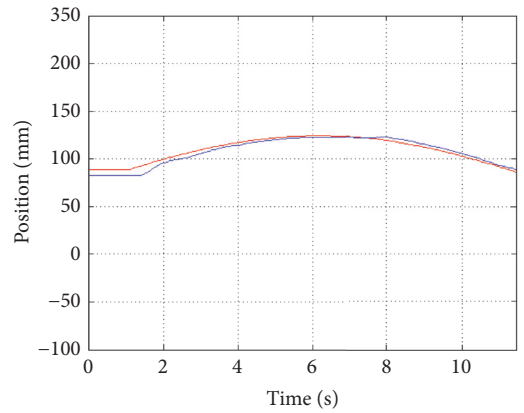
Figure 17: Cartesian movement of the MOVES command.



(a) Shaft movement X



(b) Shaft movement Y



(c) Shaft movement Z

Figure 18: Cartesian movement of the MOVEC command.

(articular, linear, and circular). For the interpolations of trajectories in Cartesian space (MOVES and MOVEC), it is necessary to calculate the IKM to find the articular values of each interpolated Cartesian point, which implies that it requires more processing in these commands, compared to MOVEJ.

It is possible to determine the workspace presented by the robot manipulator Scorbot ER_4PC, taking into consideration the mechanical and electrical limitations. It means that the maximum work space was founded, restricting the manipulator's movements due to the positions that are not allowed for this type of anthropomorphic robot.

The use of rigorous mathematical models allows us to obtain the algorithms generators of trajectories, which supply articular and Cartesian coordinates for the movement of the manipulator. In the latter case, shown in Figures 17 and 18, the real coordinates of the end effector show oscillations on the desired coordinates. These results are acceptable as they are within the subjection tolerance of the tool used for testing (clamp with radius of 32.5 mm).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] W. Xu, L. Dongsheng, and W. Mingming, "Complete calibration of industrial robot with limited parameters and neural network," in *Proceedings of the IEEE 4th International Symposium on Robotics and Intelligent Sensors (IRIS '16)*, pp. 103–108, Tokyo, Japan, December 2016.

[2] F. Petit and A. Albu-Schäffer, "Cartesian impedance control for a variable stiffness robot arm," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '11)*, pp. 4180–4186, San Francisco, Claif, USA, September 2011.

[3] P. Arena, S. Fazzino, L. Fortuna, and P. Maniscalco, "Game theory and non-linear dynamics: The Parrondo Paradox case study," *Chaos, Solitons & Fractals*, vol. 17, no. 2-3, pp. 545–555, 2003.

[4] M. S. Kazemi and M. J. Dominguez, "Simulation and evaluation of neuro-controllers applied in a SCORBOT," in *Proceedings of the IEEE International Conference on Automatica (ICA-ACCA '16)*, Curico, Chile, October 2016.

[5] E. Zurek Varela and R. López Beltrán, "Carga y descarga automática de una fresadora de control numérico utilizando un robot Scorbot-ER 4pc," *Revista Científica Ingeniería y Desarrollo*, vol. 7, pp. 113–119, 2011.

[6] V. A. Deshpande and P. M. George, "Analytical Solution for Inverse Kinematics of SCORBOT-ER-Vplus Robot," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 3, 2012.

[7] A. González Echeverri, *Análisis cinemático y dinámico del robot Scorbot-ER Vplus para la nueva configuración en una base deslizante [Bachelor's thesis]*, Universidad Tecnológica de Pereira, Pereira, Colombia, 2014.

[8] E. Robotec, *Scorbot er-4pc: User's Manual*, 1982.

[9] A. Elfasakhany, E. Yanez, K. Baylon, and R. Salgado, "Design and development of a competitive low-cost robot arm with four degrees of freedom," *Modern Mechanical Engineering*, vol. 1, no. 2, article 47, 2011.

[10] National Instruments, The CompactRIO Platform, endless Capabilities, Unrivaled Performance, 2014, http://www.ni.com/compactrio/.

[11] H. C. Fang, S. K. Ong, and A. Y. C. Nee, "Interactive robot trajectory planning and simulation using augmented reality," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 227–237, 2012.

[12] P. Corke, *Robotics, Vision and Control*, Springer, Berlin, Germany, 2011.

[13] A. Hemami, "Kinematics of two-arm robots," *IEEE Journal on Robotics and Automation*, vol. 2, no. 4, pp. 225–228, 1986.

[14] A. A. Mohammed and M. Sunar, "Kinematics modeling of a 4-DOF robotic arm," in *Proceedings of the International Conference on Control, Automation and Robotics (ICCAR '15)*, pp. 87–91, Singapore, May 2015.

[15] J. H. C. Rojas, R. R. Serrezuela, J. A. Q. López, and K. L. R. Perdomo, "LQR hybrid approach control of a robotic arm two degrees of freedom," *International Journal of Applied Engineering Research*, vol. 11, no. 17, pp. 9221–9228, 2016.

[16] C. S. G. Lee, "Robot arm kinematics, dynamics, and control," *Computer*, vol. 15, no. 12, pp. 62–80, 1982.

[17] A. Cayley, *An Elementary Treatise on Elliptic Functions*, Deighton Bell & Co, Cambridge, UK, 1876.

[18] T. Barrera, A. Hast, and E. Bengtsson, "Incremental spherical linear interpolation," in *Proceedings of the Annual SIGRAD Conference. Special Theme-Environmental Visualization*, pp. 7–10, Linköping University Electronic Press, November 2004.

[19] V. E. Kremer, *Quaternions and SLERP*, University of Saarbrucken, Department for Computer Science Seminar Character Animation, Saarbrücken, German, 2008.

[20] J. S. Ahn, W. J. Chung, and S. S. Park, Application of Quaternion Interpolation (SLERP) to the Orientation Control of 6-Axis Articulated Robot using LabVIEW® and RecurDyn®.

[21] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*, pp. 245–254, San Francisco, Calif, USA, July 1985.

[22] R. R. Kumar and P. Chand, "Inverse kinematics solution for trajectory tracking using artificial neural networks for SCORBOT ER-4u," in *Proceedings of the 6th International Conference on Automation, Robotics and Applications (ICARA '15)*, pp. 364–369, Queenstown, New Zealand, February 2015.

[23] R. R. Serrezuela, A. F. C. Chavarro, M. A. T. Cardozo, A. L. Toquica, and L. F. O. Martinez, Kinematic modelling of a robotic arm manipulator using matlab, 2006.

[24] Y. Angal and A. Gade, "LabVIEW controlled robot for object handling using NI myRIO," in *Proceedings of the IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT '16)*, Pune, India, December 2016.

[25] M. M. Ali, H. Liu, N. Stoll, and K. Thurow, "Kinematic analysis of 6-DOF arms for H20 mobile robots and labware manipulation for transportation in life science labs," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 10, no. 4, pp. 40–52, 2016.