

Educational Robot Arm Development

David Sáenz Zamarrón, Nancy Ivette Arana de las Casas, Enrique García Grajeda,
José Francisco Alatorre Ávila, Naciff Arroyo Jorge Uday

Tecnológico Nacional de México,
Instituto Tecnológico de Ciudad Cuauhtémoc,
División de Estudios de Posgrado e Investigación,
Mexico

davsaenzz@gmail.com

Abstract. A robot is a complex machine that involves the conjunction of many technologies working harmoniously together to provide to the final user, a nice interface to interact. The kind of problems need to be solved to have a robotic arm involves overcoming lateral loads, power consumption, solution of kinematics equations, etc. Peter Corke's Robotics Toolbox [1] is a computer library useful to design, model, visualize and simulate a robot and it is widely utilized in the present study. This paper describes the design process and construction of a 4-DOF robotic arm, involving CAD, CAM, electronics, and Matlab's Robotics Toolbox to solve kinematics. It constitutes a low-cost platform, in a process of permanent improvement, for the study of laboratory courses of design, manufacturing, electronics and robotics, essentials for many engineering curriculums. The platform provides the student with hands-on experience that consolidates classroom lectures.

Keywords. Educational robot, Matlab robotic toolbox, CNC machining.

1 Introduction

Robot arms are programmable electro-mechanical devices designed to carry out specific tasks such as assembly, material handling, and loading of a tool for: welding, painting, spraying, etc. To understand the complexity of robots, engineering knowledge of design, manufacturing, mechanical, electrical, computer science and mathematics are required.

Applications and developments in the field of robotics have been increasing over time and demand trained graduates who must be proficient in all the technologies related to it.

Teaching engineering courses and specially robotics, is an important subject in undergraduate and graduate school. Engineering educators agree that experience with the real world cannot be taught just in the classroom, hands-on tangible experience is needed.

In addition, it is well known that the more active and pragmatic the students are involved in applying a subject, the better the learning of its theoretical aspects. That is why laboratories are paired to theoretical classes to combine these two important learning aspects. Thus, when teaching a robotics course, it is recommended the use of an experimental platform in the learning process [2] as it allows a practical experience demonstrating the basic concepts and keeping the students' interest and motivation.

A possible platform are commercial industrial manipulator robots, but this constitute limited resources for students to access because of their high costs for institutions. In addition, they could be only used through their proprietary motion description languages, which are specific to any given manufacturer, requiring to spend long time learning any of them, and the user won't be involved in designing aspects of the robot. However, even if an industrial robotic design is trying to be built, it is still difficult to achieve, because its parts require long, and expensive processes operated in specialized laboratory equipment to generate an industrial design.

A more reachable platform could be built with open hardware/software philosophy. In this case, the robot designer must deal with concepts like rigid links which are interconnected by joints into a

serial chain and manipulated with servo motors in the so-called Joint space. Additionally, there is another Cartesian space where objects and task are defined to operate. Transformation between Joint and Cartesian space and vice versa is the issue of kinematics and it is required for any robot arm.

Hence forward kinematics is defined as transformation from joint space to Cartesian space whereas inverse kinematics is defined as transformation from Cartesian space to joint space. Solution for kinematics is available in algorithms written into specialized libraries. That is the case of the open source Matlab Toolbox for robotics: Robotics Toolbox, this has been translated into a series of different languages such as Python, SciLab and LabView.

The Toolboxes have some important virtues. First, they have been available for a long time and have been used by many people for different problems. So the code could be said to have a high degree of reliability. The Robotics Toolbox [1] is a software package that allows a MATLAB user to readily create and manipulate datatypes fundamental to robotics such as homogeneous transformations and trajectories. Functions provided, for arbitrary serial-link manipulators, include forward and inverse kinematics.

When considering moving the joints, the driver could be an open-source hardware resource. These has grown up in the last times, offering many possibilities in open platforms, and every time come new products at low cost. Open hardware offers the possibility to educators and researchers to add and program the hardware devices as they want, making the systems completely customizable. In the case of robotics, there is a tendency towards open hardware products [3] because its low cost and easy development.

Thus, the need to develop an experimental platform to serve the purpose of a permanent laboratory for universities and research institutions is justified.

To emphasize on the general principles and provide the student with both a theoretical appreciation of, and practical design and construction experience, a robot arm manipulator has been developed. It is especially valuable for many universities with limited economic resources.

Therefore, it is a good alternative for such robot because it is inexpensive to build.

This paper shows the design, manufacturing, mechanics, electronics, and software of an educational robot arm manipulator. Additionally, mathematical model for the forward and inverse kinematics problems were implemented by the educational robot arm model. A GUI software interface of great importance was developed to use the physical robot arm together with its virtual mirror.

The platform prototype is a robotic arm manipulator with 4 degrees of freedom (DOF). The arm consists of four servomotors: the base, shoulder, elbow, and the wrist, where a marker is hold as a tool. The motor control is performed by a servo controller Arduino UNO that allows a serial connection to send and receive commands to a computer.

The robot arm has been taken as a case study; it utilizes Matlab/Arduino as the tools for testing the characteristics of the robot. The developed platform is used as an educational tool. This work will continue to increase the education, training, research and development possibilities for robotics classes and research in graduate and undergraduate studies.

2 Background

Educational robotic research efforts have been reported in the literature. In [4] a robotic course was developed in Korea integrating the use of LEGO kits, humanoids, and industrial robots to improve competitions in engineering education. The author presents an approach to teaching robotics to undergraduate students that used modular, reconfigurable robots developed at LEGO. These kits are accessible in cost, permitting students to acquire experience in the kinematic design of fixed robot manipulators. But students, in fact, did not build the robots.

A commercial 5 DOF robot with the CRS Catalyst-5 from Thermo Fisher Scientific Inc. was controlled through a Matlab/Simulink open-architecture interface, being mostly a software project [5]. With [6], an educational robotic arm was built, but the interface to it, is through

Matlab/Simulink, so the robot can be used just for experimented programmers.

Others educational robotic arm were thought as remote laboratories developing e-learning potentialities in the field of robotics tele laboratories. A virtual training environment was created through a commercial SCARA-type AdeptOne-MV robot arm, a video camera, and an internet connection [7].

A tele laboratory was also created to access remotely a SCORBOT ER-V PLUS robot based on Web by using the free open-source software: Scilab, Comedi and Linux [8]. In the two previous projects, the students acquired design ability by doing active learning, in a tele laboratory characterized by high immersivity. Virtual laboratories were accessible 24 hours a day and allowed the students to practice their robotics skills at home and at their free time. However, students were focused more on training than in robotic design.

The virtual robotic systems represent an illustrative and cost-effective solution. However, such systems, being mostly soft, may not offer the necessary exposure corresponding to the real robot performance. Virtual models do not address the complexities such as backlash, friction, non-collocation, etc. associated with the physical systems. Typically, the accuracy and credibility of results obtained in a simulated environment are not comparable with physical experiments.

Additionally, training platforms based on commercial robots and development of software had been reported. In [9] the GUI software for kinematics of a commercial 5 DOF Lynx-6 was developed. Visual Studio.Net 2005 was used for the implementation. Students learn how to use knowledge and techniques to carry out a set of standard tasks for robots.

A robotic arm was used to test the sliding mode and computed torque control strategies. But no detail on the construction of the robot was given, neither an educational aspect of the project was especially underlined [10].

BRACON robotic arm based on Dynamixel servomotors and Python software was developed [11]. The project presented a low-cost robotic arm with a nice GUI, but important details of links mechanical design were not shown.

An industrial robotic arm was taken to be modeled using D-H methodology and to generate simulated trajectories based on computer vision techniques [12]. In more hands-on approaches, students can develop their own robot structure and then program it with a language. There have been found several papers addressing different areas of teaching and learning with robots for the development of suitable teaching curricula.

A 6 DOF robotic arm was built and reported in [13]. Servomotors were used, details of the design were given, a nice GUI was created, but no solution of the inverse kinematic problem was done. A particularly good effort to amplify access to practical experiences in educational robotics was done in [14]. The activities implemented with this platform included: playing, building, exploring, programming, competing, and learning various science subjects. However, it was completely centered in Lego Mindstorm kits, which is not adequate for university courses and research because of its limited hardware. In addition, no fabrication processes skills were developed in that project.

There are some papers documenting the use of many different low-cost open-source microcontroller cards based on ARM processors. Given that the main purpose of these works was to build a low cost open-source platform to introduce robotics concepts, any card could be suitable since high computational resources were not required. However, its programming should had been simple because maybe the students did not know much about programming or perhaps, they were learning it.

A good review of low-cost open hardware microcontroller boards was given in [15]. Arduino, Raspberry Pi and BeagleBone among others were mentioned. These boards are useful to control educational platforms like Lego. Arduino support for Simulink software was proposed to control the hardware. In [16] an ED722C 6-DOF commercial robotic arm was used to be motion controlled through a camera seeing objects in the robot's workspace.

The project was more interested on the technical aspects of the tasks development instead of getting educational goals. In [17] Arduino was used to control several types of robots like line

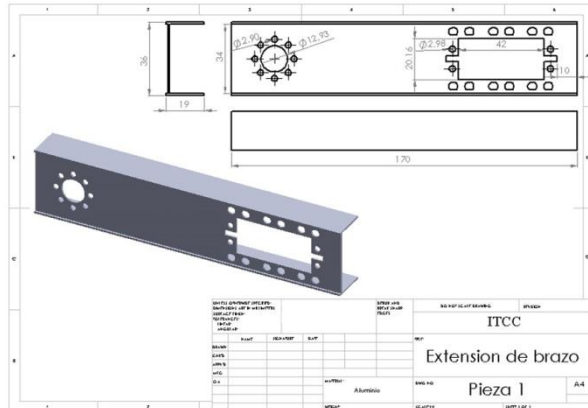


Fig. 1. Mechanical flat of aluminum framework

followers, bug-like, boat-sailer, explorer and battle oriented.

Arduino arose in 2005 and due to its low cost and open hardware policy, in a few years it has achieved widely spread within the scientific and research community as the brain of various kind of projects.

The Robotics Toolbox [1] provides a diverse range of functions to simulate arm type robots. The original Toolbox started in 1990 and was dedicated only to arm type robots and supported a very general method of representing the structure of a manipulator with serial joints using objects of Matlab. Arbitrary manipulators with serial links can be created and the Toolbox provides functions for direct and inverse kinematics and dynamics.

The Toolbox includes functions to manipulate and convert between data types such as vectors, homogeneous transformations and unitary quaternions which are necessary to represent position and three-dimensional orientation.

Other Matlab toolbox for teaching, modeling and simulation of robotics manipulators had been developed like [18]. This toolbox has been called ARTE (A Robotics Toolbox for Education). Another Matlab toolbox had been developed; it is called ROBOLAB and includes into its library the 16 different 6-DOF fundamental serial robot manipulators [19].

A very convenient GUI allowed the movement of virtual robots chosen in the software. Both toolboxes: ARTE and ROBOLAB, by far, are not so deployed and accepted like the Corke's one [1].

3 Design and Construction

Through 3D CAD software for mechanical modeling: SolidWorks, the basic link was created by means of a draw with real measurements. This is done in order to later build and assemble each of them together with its servomotor and, with four of them, form the articulations and links, needed for the construction of the final assembly.

The development of the piece in software allows seeing the characteristics that the element has before manufacturing it, this facilitates finding errors and making the corrections that correspond. Two terminals were designed for each link that contains the assembly until the end-effector.

The link (figure 1) that was designed has two terminals, each one on one end and with a different function. The one on the right has a rectangular hole that holds the servomotor to the link with screws.

The one on the left, has the function of connecting with the next link in the kinematic chain. Figure 1 shows the design of the link on an aluminum framework that is the material selected to make the piece.

The circular connection, leftmost connection on figure 1 is connected to the next link, through a servo hub, which runs perfectly through a servo block kit. The servo hub is made of aluminum, it has a length of 1", a hub diameter of 1", a shaft diameter of 0.5", and utilize the 0.77" hub pattern (figure 2).

The aluminum framework has ends bent at 90°, which allows the piece to function as a reinforcement structure. In addition, the material is light, resistant and easy for machining. There was access to a CNC milling machine DYNA MACH EF8035 to perform the machining by means of chip cutting by turning a 3/8" cutter with 4 cutting edges.

Through milling, it is possible to machine the most diverse materials, such as wood, steel, cast iron, non-ferrous metals, and synthetic materials, on flat or curved surfaces.

The case of aluminum is particularly soft when cut. In addition, the milled pieces were roughened and tuned to avoid roughness. To operate the CNC milling machine, a program in G code was wrote, this is the most used in this type of numerical control machines.

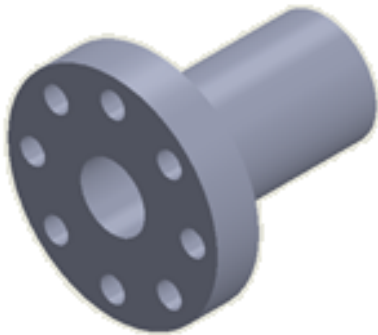


Fig. 2. Servo hub

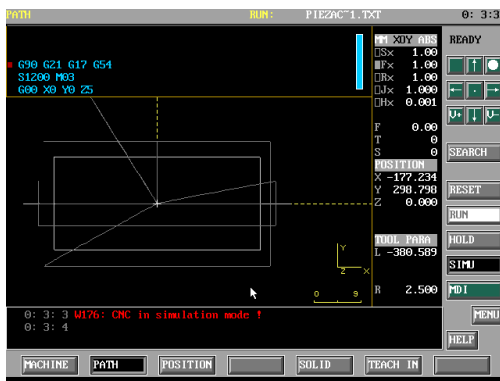


Fig. 3. Rectangular cut simulation

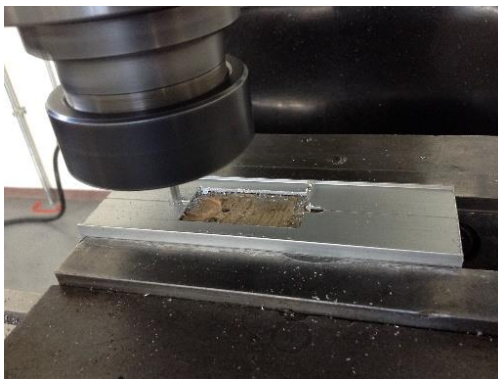


Fig. 4. Machined of link, servo edge

In general terms, G-code is a language by means of which computer-controlled machines and tools can be told what to do and how to do it, for this machine, the 4M software was used. These actions are defined by instructions on where to move, how fast to move and what path to follow. The G codes can be those that contain movements

and fixed cycles and M, those that are auxiliary functions. Next, the simulation stage comes, it helps the designer of the piece to foresee and detect problems related to the design before it even physically being built. Hence offering a much more efficient development stage in terms of time and cost.

Figure 3 shows the simulation of the machining of the link at the end of the servomotor. During the process of the simulation, it is possible to observe by which places the cutter is passing and, in this way, some possible error can be found. It is also possible to see in which parts it is, where the milling machine is cutting and in which it is not, this by means of the color of the lines that the simulation leaves. If they are marked lines it is when it is cutting and those that are weaker color is when it is just making a quick movement.

Figure 4 displays the milling of the link at the end of the servomotor. Figure 5 demonstrates the simulation of the milling of the link at its connecting end to the next link. The physical machining of the link is performed at the connection end to the next link, figure 6.

The material that is machined is the 1mm thick aluminum framework with irrelevant measures since the code was designed so that this does not matter. That is, the machining starts only from a center and from there it opens. The proper speed and rotation speed must be specified to obtain adequate machining.

To increase a servo's load-bearing capabilities is necessary to isolate the lateral load from the servo spline and case. It can be done by using a servo block. The servo block allows users to create complex, extremely rigid, structures with ease using standard Hitec servos. Using a clamping hub on the 1/2" aluminum servo shaft provides an easy way to adjust the position of the component attached to the ServoBlock.

The robust aluminum framework acts as a servo exoskeleton, greatly enhancing the mechanical loads the servo can withstand. The .77" hub pattern is repeated throughout the framework to allow endless attachment options. It is compatible with all standard size Hitec servos. Figure 7 displays a Hitec servo attached to a servo block kit. The robot arm rotational joints are controlled by dedicated servo motors. Servomotors are some kinds of traction elements.

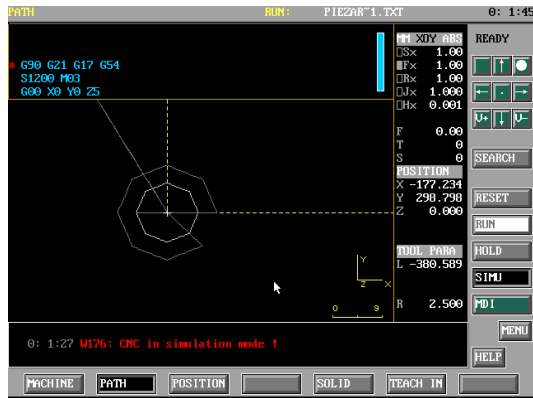


Fig. 5. Circular cut simulation

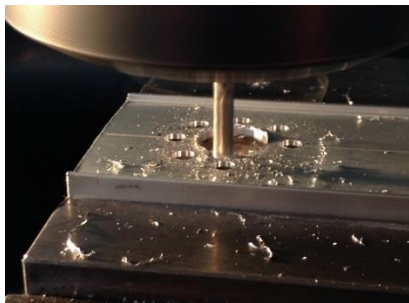


Fig. 6. Machined of link, next link connection edge



Fig. 7. Servo block kit and servo

The servo selected was the powerful Hitec HS-625MG. This high-quality servo is perfect for

mechatronic and robotics needs. This element is one of the strongest metal gear servos that is available and at a low cost. The HS-625MG servo comes standard with a 3-pin gold plated power and control cable. The HS-625MG can take in 6 volts and deliver 94 oz-in of maximum torque at 0.15 sec/60°.

The servomotors have an acceptable accuracy, given in milliseconds. They can be manipulated by any microcontroller with PWM module, in this case, the Arduino UNO. The microcontroller for example, produces a pulse width, this is sent into a servomotor input, and it can respond with a movement given in degrees.

In this approach being described, four servo motors are controlled to form the kinematics chain. For the robot arm being developed, the base link is the one that most care must be taken, because of the lateral loads it is imposed. A swivel hub connection was used on the base link to allow a 360° horizontally rotation.

The swivel hub lets the connection of any two parts together and has them swivel a full 360°. Internal ball bearings allow for smooth motion. Four holes are on both sides, making attachment easy. Figure 8 shows the swivel hub.

However, the servo motors draw considerable power and must be fed with enough power supply. For this purpose, a 18W 6-in-1 based on the LM350K voltage regulator was built. It provides 6 VDC @ 3A, 18W max output. It takes a 110VAC, 50/60Hz input. Figure 9 illustrate the power supply and Arduino UNO servo controller.

Servo motors have three wires: power, ground, and signal. The servo's power wire should be connected to the 6V pin on the power supply. The ground wire should be connected to a ground pin on the power supply and the Arduino board. The signal pin should be connected to a digital pin on the Arduino board. To facilitate these connections special 3-feet long cable was built; it is seen in figure 10.

The final mechanical assembly of the prototype of the real robot along with all its wiring and the end-effector, a marker, is revealed in figure 11. This building stage allows students to get acquainted to various aspects of technical design, teamwork, and material constraints. In this stage, design errors frequently occur like building too big links, selection of weak motors, structures stressed



Fig. 8. Swivel hub



Fig. 9. Power supply & controller



Fig. 10. Special connection cable

because of its own weight or selection of an inappropriate material. These errors had to be overcome to have an operative arm. Figure 12 displays the serial communication scheme between the computer, which is using MATLAB® / Robotics Toolbox [20] and the Arduino; as well as the communication by train of pulses of the Arduino (library Servo) and the servomotor.

A sequence of four consecutive integer values is sent to serial servo controller from the computer. These are a sync byte (\$), the desired position joint for each of the four servos (q_i) and the final character (#). Rotational position of each of the servo motors is determined by the specific angle (q_i) by an internal servomotor's closed loop feedback.

4 Modelling

The kinematics defines the movement of a body as the transformation from its Cartesian space to its joint space and vice versa [21].

A manipulator with serial joints contains a chain of joints and mechanical links. Each joint can move the league of its neighbor further away from its nearest neighbor to the origin. One end of the chain, the base, is fixed and the other end is free to move in space, the latter holds the end-effector. The kinematic modeling of a robot is categorized into forward kinematics and inverse kinematics.

The forward kinematic problem consists of determining the location of end-effector in the work space, i.e., position and orientation with respect to a coordinate system that is taken as reference. It is determined based on the joint variables and the parametric values of the robot configuration given by its D-H parameters [22]. The inverse kinematics problem refers to finding the values of the joint variables that allows the manipulator to reach the given location [23].

For a manipulator with N joints, numbered from 1 to N numbered. Link 1 is the base of the manipulator and link N loads the end-effector. The joint i connects the link $i-1$ to the link i and therefore the joint i moves the link $i-1$. A link is a rigid body that defines the spatial relationship between two neighboring joints. A link is specified by two parameters, its length a_i and its rotation α_i . The joints are also described by two parameters.

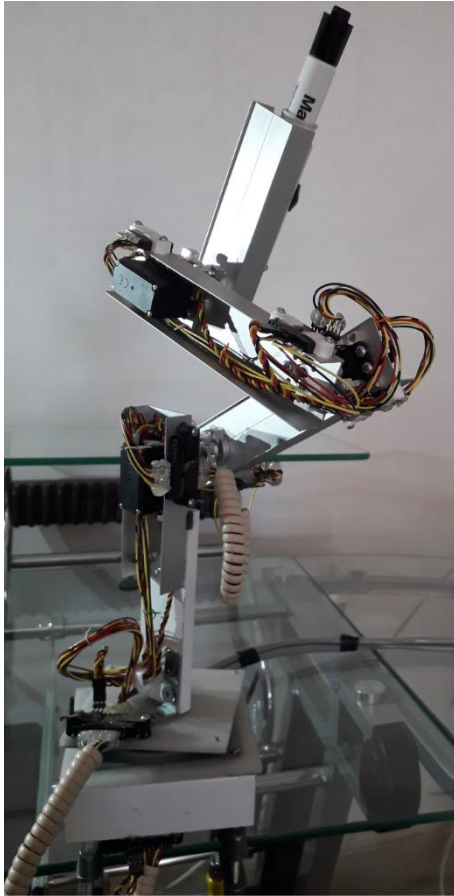


Fig. 11. Robot on rest position

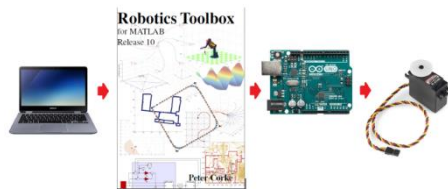


Fig. 12. Robot communication scheme

The displacement of link d_i is the distance from one joint to the next along the axis of the joint. The angle of the joint θ_i is the rotation of one joint with respect to the next around the axis of the joint. Transformation link matrix is shown in (1):

$${}^{i-1}T(\theta_i, d_i, \alpha_i, \alpha_i) = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & \cos \alpha_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & \alpha_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

4.1 D-H Parameters

The structure of articulations for the robot is described by a chain of joints of Revolution. A systematic way of describing the geometry of a chain of serial joints was proposed by Denavit and Hartenberg and is implemented into the Matlab's Robotics Toolbox [1].

Next, Table 1 shows the Denavit-Hartenberg parameters for the robot developed. The following segment of code shows variables defining the D-H parameters into the Matlab program:

```
N=5; %4 DOF
% D-H Parameters
Theta =[q(1,1) q(1,2) q(1,3) q(1,4) q(1,5) q(1,6) ];
d      =[12  0  0  0  10  0  ];
a      =[0  12  12  0  0  0  ];
alpha  =[-pi/2  pi  -pi/2  0  0  ]
```

4.2 Forward Kinematics

Forward kinematics is expressed as the pose of the end-effector of the robot arm according to the values of the joints. Homogeneous transformations are used; these are simply the product of the transformation matrices of each individual joint.

A manipulator robot with six articulations or degrees of freedom (DOF) allows reaching an arbitrary pose in the end-effector. The transformation of the manipulator is written as 0T for the 6-axis robot [24, 25].

The first three columns in the matrices represent the orientation of the end effectors, whereas the last column represents the position of the end effectors.

The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the end-effector.

Forward Kinematics equations are generated from the transformation matrixes (4)-(9) and the forward kinematics solution of the arm is the product of these six matrices identified as 0T .

The approach divides the forward kinematics problem in two parts. From 0T to 2T is the elbow manipulator part and from 3T to 5T the wrist movement description. Elbow manipulator and wrist are defined through D-H methodology.

Table 1. D-H Parameters, angles (Rad), lengths (cm)

Link	θ_i	d_i	a_i	α_i	offset
1	q_1	12	0	$-\pi/2$	$.222\pi$
2	q_2	0	12	π	1.22π
3	q_3	0	12	π	1.33π
4	q_4	0	0	$-\pi/2$	$.888\pi$
5	q_5	10	0	0	0
6	q_6	0	0	0	0

The direct model is the relation that allows determining the matrix column x of operational coordinates of the robot corresponding to a given configuration q , see (2):

$$x = f(q). \quad (2)$$

The direct geometric model of a robot can be obtained from the homogeneous transformation matrix of the robot that defines the N links of the terminal link with respect to the link 0 of the base of the robot. In the case of simple structure of robots, the transformation matrix is given by (3):

$${}^0T = {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6. \quad (3)$$

Equations (4-9) are the transformation matrices for each link:

$${}^0T_1 = \begin{bmatrix} Cq_1 & 0 & -Sq_1 & 0 \\ Sq_1 & 0 & Cq_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4)$$

$${}^1T_2 = \begin{bmatrix} Cq_2 & Sq_2 & 0 & a_2 Cq_2 \\ Sq_2 & -Cq_2 & 0 & a_2 Sq_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

$${}^2T_3 = \begin{bmatrix} Cq_3 & Sq_3 & 0 & a_3 Cq_3 \\ Sq_3 & -Cq_3 & 0 & a_3 Sq_3 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6)$$

$${}^3T_4 = \begin{bmatrix} Cq_4 & 0 & -Sq_4 & 0 \\ Sq_4 & 0 & Cq_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (7)$$

$${}^4T_5 = \begin{bmatrix} Cq_5 & -Sq_5 & 0 & 0 \\ Sq_5 & Cq_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (8)$$

$${}^5T_6 = \begin{bmatrix} Cq_6 & -Sq_6 & 0 & 0 \\ Sq_6 & Cq_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (9)$$

The code below is the calculation of the forward kinetics made in the project using the Corke's *fkine* Toolbox method [1]:

```

qb=handles.Robotqb;
q=qb(index,:)*pi/180;
qi=q;
Point=Cl.fkine(qi); %Position in world coordinate system
handles.Matriz=Point;

```

4.3 Inverse Kinematics

Inverse kinematics is a procedure that seeks to obtain the required joint coordinate values q , given the desired Cartesian pose of the end-effector x . This is a more difficult problem than forward kinematics. On the contrary of forward kinematics, there exist multiple solutions in inverse kinematics [26, 27]. Some constraints can be used to decrease the number of solutions for simplicity. Into Corke's Robotic Toolbox [1] there are two methods useful to solve the inverse kinematics problem:

- Closed-Form solution (algebraic and geometric methods).
- Numerical solution.

For the first one the kinematics equations that relate the joint variables to the end-effector pose are nonlinear and the Toolbox gives only a restrained description of the manipulator in terms of kinematic parameters. The robot must have 6-axis; also, the robot must have the three wrist axes intersect at a single point (spherical wrist).

For the case of robots, which do not meet this specification, an iterative and numerical slower solution is implemented.

The numerical solution takes advantage of the differential equations that relates joint rates to end-

effector velocities, which are linear. It takes the form of (10), which can be easily solved but produces joint rates, rather than the joint values themselves:

$$\mathbf{V} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \quad (10)$$

where $\mathbf{q} = [q_1, q_2, \dots, q_n]^T$ is the joint variable vector and $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n]^T$ is the joint rates vector and produces spatial velocities $\mathbf{V} = [v_x, v_y, v_z, w_x, w_y, w_z]^T$.

In order to track a trajectory that can be described by a sequence of desired pose matrices at regular time intervals t_k along with desired end-effector velocities $\mathbf{V}(t_k)$ the joint rate vectors can be computed based on (11):

$$\dot{\mathbf{q}}(t_k) = \mathbf{J}[\mathbf{q}(t_k)]^{-1}\mathbf{V}(t_k). \quad (11)$$

Using the numerical integration approximation shown in (12) joint variable vector for the next point on the trajectory is calculated.

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \dot{\mathbf{q}}(t_k)\Delta t, \quad (12)$$

Δt is the time interval between points on the trajectory.

It should be small enough to minimize computation errors from one step to the next on the trajectory. The kinematic inversion, based on the manipulator Jacobian, is applicable to robots of any architecture but requires a nonsingular Jacobian matrix at every point on the trajectory, which is rank deficient compared to the maximum of the Jacobian, singularity is stated in (13).

$$|\mathbf{J}(\mathbf{q})| = 0. \quad (13)$$

At the singularity, joint rates will go to infinity; but even if the robot is close to a singularity, there is the problem for some Cartesian end-effector velocities that require extremely high joint rates. An alternative for this problem is to use the Jacobian transpose.

The Jacobian transpose transforms a wrench applied at the end-effector to torques and forces experienced at the joints, see (14):

$$\mathbf{Q} = \mathbf{J}^T(\mathbf{q})\mathbf{g}, \quad (14)$$

where \mathbf{g} is a wrench which is a vector in the world coordinate frame of forces and moments.

$$\mathbf{g} = [f_x, f_y, f_z, m_x, m_y, m_z].$$

\mathbf{Q} is a joint force vector. The elements of \mathbf{Q} are joint torque or force for revolute or prismatic joints, respectively. This mapping from the wrench to joint forces differs from the velocity because singularity will not happen, as it can be for velocity, since the Jacobian's transpose is used.

This property is harnessed to solve the inverse kinematic problem into the Corke's Toolbox numerically. The Toolbox develops this approach based on the forward kinematics and the Jacobian transpose which can compute for any manipulator configuration, these functions have no singularities. Into the Toolbox this approach assumes a special spring between the end-effector of the different poses which is pulling and twisting the robot's end-effector toward the desired pose with a wrench proportional to the difference in pose. The robot has virtually viscous dampers so the joint velocity due to the applied forces will be proportional (15):

$$\dot{\mathbf{q}}(t_{k+1}) = \mathbf{Q}(\mathbf{q})/B, \quad (15)$$

where B is the joint damping coefficient. The update for the joint coordinate is the same as (12).

This algorithm is implemented into the *ikine* Toolbox method [1] and is used in this project in the fragment of code below.

There, an incremental pose kinematics inverse is used in order to have world (x, y, z) displacements by using the same Rotation submatrix $\mathbf{R}_{3 \times 3}$, and varying Translational submatrix $\mathbf{T}_{3 \times 1}$ into the Homogeneous Transformation matrix:

```

Point=handles.Matriz;
Point(windice,4)=Point(windice,4)+w;
%handles.Matriz=Point;
%Obtaining the value of links for the desired point
q=Cl.ikine(Point,qi,[1 1 1 1 0
0], 'pinv', ... 'ilimit', 2000, ...
'alpha', .05, 'tol', .3);
if (isequal((q<0)|(q>pi), zeros(1, NArt)))
qb(index,:)=round(q*180/pi);
handles.Matriz=Point;
else
q=qj;
end
handles.Robotqb=qb;

```

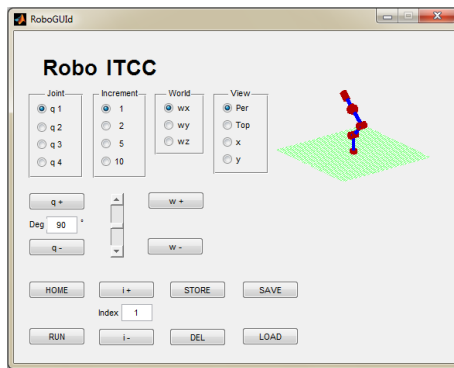


Fig. 13. Graphical Interface

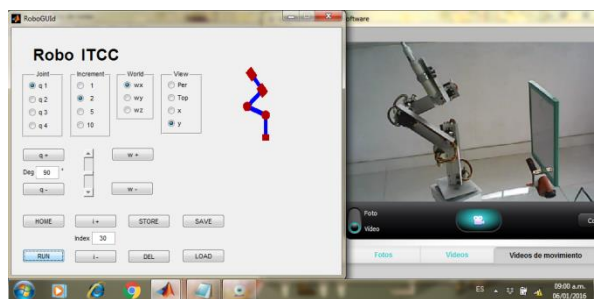


Fig. 14. Virtual and real robot arm

5 User Graphics Interface

The need for a graphical interface (GUI) is very important because it requires a user-friendly interface for users who have limited programming knowledge.

It is also required that GUI be intuitive, in such a way that anyone familiar with a computer can easily handle the robotic arm just using buttons, text boxes, sliders, etc., and observe changes in both virtual and physical robot arm.

The graphical interface designed for the four degrees of freedom robot (figure 13) was programmed using the MATLAB® software and presents a compact version compared with interfaces developed in other projects, such as [28].

The interface that was created has four controls (buttons: $q+$, $q-$, edit box: Deg, and a slide bar) that modify the angle of one of the four joints that is specifically selected to be changed (Joint). It constitutes the direct kinematics. In addition, this modification of the angle is changed in one of four

intervals given by the selection that is operated in *Increment*.

The selection that affects the articulation space (*Joint*) also affects the *World* space (inverse kinematics) by selecting one of the three references (w_x , w_y , w_z) for an incremental movement. There is a selection to visualize the virtual robot from four different points of *View*.

The *Home* button sends the robot to its resting position. The *Store* option stores a joint position in memory and increases the *Index*. The *Del* button removes the joint position indicated by *Index*.

The *Save* button saves all the joint positions of the robot to a *.mat file, thus recording a trajectory. A trajectory is a spatial construction, a location in space that leads from an initial position to a final position. The *Load* button loads a *.mat file with a path to the robot. The *Run* button goes through the joint positions that constitute the path to be traveled.

When one of the joint controls is activated, the change produced is observed graphically in the virtual robot. Similarly, the updated value of the joint variable can be seen in the *Deg* edit box. The position of the slide bar for the modified joint is also updated. Table 2 describes in detail each of the controls of the developed graphical interface.

When the virtual robot in the software moves, the same program sends the joint positions (q_1 , q_2 , q_3 , q_4) to the controller (Arduino UNO) through the serial port so that these changes can be seen in the servo motors of the real robot (figure 14).

5.1 Programming

The calculated q_i values should be supplied into the servo controller card from the computer. The Arduino board handles the relevant angular motion to every servo motor in the robot arm. Arduino IDE is used directly to perform the programming on the board.

To establish communication between the interface and the robot, a serial communication port is used. The pseudocode that was loaded on the Arduino UNO card is presented; this code receives articulated positions from the computer and transfers them to the servo motors of the robot:

- a) Inclusion of the Servo library.

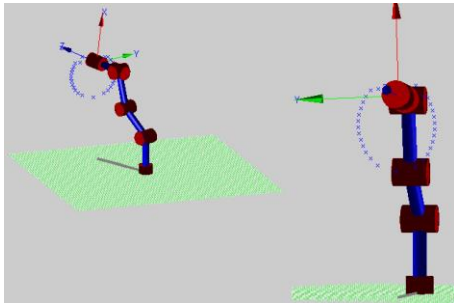


Fig. 15. Evaluation of generated positions

Table 2. Controls of the interface

Control	Description
	Select the specified joint to be rotated.
	Select the specified angular or linear increment to be moved.
	Select the specified world frame of reference to be moved.
	Select the specified point of view for the robot.
	Controls that change the angular position of the selected joint.
	Controls that change the selected world frame by a specified increment.
	Sends the robot to Home.
	Motion over all the saved positions.
	Index changes automatically when a position is stored or deleted
	Store a position
	Delete a position
	Save a trajectory to a file
	Load a trajectory from a file

- Initialization of variables and position of servomotors $q_i = [0^\circ \ 90^\circ \ 0^\circ \ 0^\circ]$.
- Read the joint positions from the serial port.
- Map the positions and write them to servomotors.
- Return to step c).

5.2 Algorithm

Next, the pseudocode of the program developed in MATLAB® [20] is presented, with the purpose of having a general understanding of its operation:

- Initialization of program variables.
- Initial joint position $q_i = [0^\circ \ 90^\circ \ 0^\circ \ 0^\circ]$.
- Opening of the serial port.
- Check user changes in controls that affect joint space, update variables and interface, solve direct kinematics.
- Check user changes in controls that affect the World space, update variables and interface, solve inverse kinematics.
- Write the current position q_i to the serial port. Under the format: \$ $q_1 \ q_2 \ q_3 \ q_4 \ #$
- Go back to step d).
- If the program is closed, close before the serial port.

6 Results

After mathematical modeling (D-H) and kinematic solutions were generated and implemented by the software, a Robot arm was developed and tested in this study.

Several trajectories were generated, one of them with the shape of a full circle (figure 15), this was done in order to evaluate the adequate generation of joint positions obtained by solving the direct and inverse kinematics equations by Matlab's Robotics Toolbox.

7 Discussion

The realization of this project allowed, since the beginning, the enrichment of a senior-level undergraduate robotic course showing many

positive results. Students were very enthusiastic because their exposition to a real case of study, also they learn to work in teams in the design and construction of the robot. The students divided tasks among themselves and scheduled their work.

The course was organized in accordance with specific goals. The first goal was to design the robot, CAD was used to design the main mechanical pieces under certain specifications, and such was the case of the robot's links.

The second goal was to develop manufacturing processes, where the students were challenged by machining pieces with Computer-Aided Manufacturing (CAM) software/machines. In addition, they constructed mechanical and electrical assemblies of the manipulator robot using tools such as band saws, drills, soldering irons, and multimeters.

The third goal was to integrate mechanism, electronic, and computer control, carrying out the achievement of a functional robotic prototype. Here the spectrum of topics required for developing the actual robotic device were components like mechanisms, structures, actuators, drivers, power sources, wiring, microprocessors, peripherals, and real-time programming. Because of the emphasis upon integration, the project centered on laboratory experiences in which student configured, designed, and implemented mechatronic devices.

The fourth goal was the implementation of kinematics. Here concepts were exposed starting from mathematical foundations, such as matrix algebra and trigonometry. Students were introduced to the concepts of direct and inverse kinematics, singularity definition and avoidance, joint rotations, translations, and trajectory computation. Furthermore, in the fifth goal, students demonstrated kinematics concepts developing and programming laboratory experiments with working robotic arm, with the programming environment.

To make the laboratory experience more like real, the six goal was devoted to put the complete experimental robotic system together and to make it move. Students established serial communication

between computer and microcontroller's embedded communication library.

Students wrote scripts in Matlab language inserting functional and structural parameters; homogeneous transformations, D-H, calculation of position and orientation of the tool, direction of the joint rotation in the kinematics model, which must coincide with the rotation of the robot's servo motor. Evaluation the extent of the workspace by varying the joint angles through the reachable space. Scripts should consider joint limitations, and thus joint angles into both, the arm controller and script, must be inside of the physical limits. Therefore, testing of robustness of the code was developed.

Then, the final goal consisted of the script ran in Matlab environment. Students did a final check of the manipulator behavior; they had some specific task to do with the robot. Students programmed the robot to follow a planned path, stored in a *.mat file; as the motion was performed, a draw was painted by a marker on a board.

A static obstacle in the robot path was put, introducing complexity to the task. From that experiment, students understood concepts such as posture (high and low elbow) and its connections with options chosen during the computation of the inverse kinematic. The previous task was repeated, but including a field of obstacles, this had repercussions on the path, precluding the robot from moving into areas previously specified as areas to avoid.

All goals of curriculum were reached, and students took advantage of the robotic course very much. However, existing opportunity areas were identified for further improvement of the robot design.¹

8 Conclusion

The objective was achieved, this is that a low-cost educational and experimental manipulator robot arm of 4 degrees of freedom was designed and built; It took advantage of the most current open hardware and software technology. It can be used in graduate and undergraduate robotic courses to

¹ Video Available: <https://home.mycloud.com/action/share/7387d7d0-56eb-4dac-a0a4-c3b880b0d57e>

realize didactically the relationships between theoretical and practical aspects of robot manipulator motions in real time. The article described the most relevant details as practical laboratory exercises of the mechanical structure, the electrical and electronic aspects, and the software developed for its management.

Thus, the design of the links was made through SolidWorks. These were machined in a CNC milling machine programmed with G codes. Mechanical exo-skeletal structures were added to reduce the effect of lateral loads on the links. The final assembly was carried out together with the wiring, power supply and controller for the physical robot. The robot was modeled mathematically using the Robotics Toolbox, solving direct and inverse kinematics and an attractive and intuitive GUI interface was created in Matlab.

The controller program was loaded with serial communication and the Servo library of the Arduino. Several tasks (trajectories) were carried out to evaluate the performance of the prototype, resulting in an adequate functioning. All these laboratory exercises provided invaluable hands-on experience that complemented classroom lectures, which are essential for many other aspects of robotics education.

Students showed enthusiasm in the application of the concepts discussed during the lectures, solving the problems by teamwork. In the future, our goal for the robotics educational platform is to integrate formally the prototype with the entire robotics curriculum, increasing the number and type of laboratory exercises routinely conducted by students of robotics track. Therefore, the platform finds its higher potential applications in educational, academic, and research sectors, representing a very encouraging starting point to further development of these activities.

Acknowledgments

This work has been funded by Tecnológico Nacional de México under the grant 6678.18-P.

References

1. **Corke, P. (2013).** *Robotics, vision and control fundamental algorithms in matlab®*. Springer.
2. **Greenwald, L., & Kopena, J. (2003).** Mobile robot labs. *IEEE Robotics Automation Magazine*, Vol. 10, No. 2, pp. 25–32. DOI: 10.1109/MRA.2003.1213613
3. **Fisher, D. & Gould, P. (2012).** Open-Source Hardware Is a Low-Cost Alternative for Scientific Instrumentation and Research. *Scientific Research an Academic Publisher*, Vol. 1, No. 2. DOI: 10.4236/mi.2012.12002.
4. **Jung, S. (2013).** Experiences in developing an experimental robotics course program for undergraduate education. *IEEE Transactions on Education*, Vol. 56, No. 1, pp. 129–136. DOI: 10.1109/TE.2012.2213601.
5. **Wood, R. (2008).** Robotic manipulation using an open-architecture industrial arm: a pedagogical overview. *IEEE Robotics & Automation Magazine*.
6. **Krasnansky, P., Toth, F., Villaverde, V., & Rahal, B. (2013).** Basic laboratory experiments with an educational robotic arm. *IEEE International Conference on Process Control*, pp. 510–515. DOI: 10.1109/PC.2013.6581462.
7. **Tzafestas, C. & Palaiologou, N. (2006).** Virtual and remote robotic laboratory: comparative experimental evaluation. *IEEE Transactions on Education*, Vol. 49, No. 3, pp. 360–369. DOI: 10.1109/TE.2006.879255.
8. **Balestrino, A., Caiti, A., & Crisostomi, E. (2009).** From remote experiments to Web-based learning objects: an advanced telelaboratory for robotics and control systems. *IEEE Transactions on Industrial Electronics*, Vol. 56, No. 12, pp. 4817–4825. DOI: 10.1109/TIE.2008.2006941.
9. **Baki, K. & Mehmet, G. (2007).** Software development for the kinematic analysis of a Lynx 6 robot arm. *International Journal of computer, electrical, automation, control and information engineering*, Vol. 1, No. 6, pp. 1575–1580. DOI: 10.5281/zenodo.1075392.
10. **Raza, U.I., Jamshed, I., & Qudrat, K. (2014).** Design and comparison of two control strategies for multi-DOF articulated robotic arm manipulator. *Control Engineering and Applied Informatics*, Vol. 16, No. 2, pp. 28–39.
11. **Rivas, D., Alvarez, M., Velazco, P., Mamarandi, J., Carrillo, J.L., Bautista, V., Galarza, O., Reyes, P., Erazo, M., Pérez, M., & Huerta, M. (2015).** BRACON: control system for a robotic arm with 6 degrees of freedom for education systems. *IEEE Proceedings of the 6th International Conference on Automation, Robotics and Applications*, pp. 358–363. DOI: 10.1109/ICARA.2015.7081174.
12. **Medrano, J., Zendejas, I., Sánchez, E., & Jara, R. (2016).** Modelado, trayectorias y control de un

- brazo manipulador industrial para la elaboración de distintas tareas por medio de visión. *Congreso Internacional de Ingeniería Electrónica ELECTRO*. Vol. 38, pp. 116–121.
13. **Fernandez-Ramírez, A., Rodríguez, R.A., & García, R. (2017)**. Modelado, diseño y construcción de un robot didáctico de 6 GDL. *Congreso Internacional de Ingeniería Electrónica. ELECTRO*. Vol. 39, pp. 229–234.
 14. **Galvan, S., Botturi, D., Castellani, A., & Fiorini, P. (2006)**. Innovative robotics teaching using Lego® sets. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 721–726. DOI: 10.1109/ROBOT.2006.1641795.
 15. **Soriano, A., Marín, L., Vallés, M., Valera, P., & Albertos, P. (2014)**. Low cost platform for automatic control education based on open hardware. *Proceedings of the 19th world congress, the International Federation of Automatic Control*, Vol. 47, No. 3, pp. 9044–9050. DOI: 10.3182/20140824-6-ZA-1003.01909.
 16. **Sarah, M., Raza, U.I., Aayman, K., Abdul, S., & Jamshed, I. (2014)**. An open-source multi-DOF articulated robotic educational platform for autonomous object manipulation. *Robotics and Computer-Integrated Manufacturing Elsevier*, Vol. 30, pp. 351–362. DOI:10.1016/j.rcim.2013.11.003.
 17. **Warren, J.D., Adams, J., & Molle, H. (2011)**. *Arduino Robotics*. Apress.
 18. **Gil, A., Reinoso, O., Marin, J., Paya, L., & Ruiz, J. (2014)**. Development and deployment of a new robotics toolbox for education. *Wiley Periodicals*, Vol. 23, pp. 443–454. DOI:10.1002/cae.21615.
 19. **Kucuk, S. & Bingul, Z. (2009)**. An off-line robot simulation toolbox. *Wiley Periodicals*, Vol. 18, pp. 41–52. DOI:10.1002/cae.20236.
 20. **MathWorks (2019)**. *Matlab, Release Notes*. [Online] Available: <http://www.mathworks.com>
 21. **Manseur, R. (2006)**. Robot Modeling and Kinematics. *Da Vinci Engineering Press*.
 22. **Fu, K.S., Gonzalez, R.C., & Lee, C.S.G. (1987)**. *Robótica: Control, Visión e Inteligencia*. McGraw Hill.
 23. **Barrientos, A., Peñín, L.F., Balaguer, C., & Aracil, R. (1997)**. *Fundamentos de Robótica*. Mc Graw Hill.
 24. **Spong, M.W. & Vidyasagar, M. (1989)**. *Dynamics and Control*. John Wiley & Sons.
 25. **Craig, J.J. (2006)**. *Robótica*. Pearson Prentice Hall.
 26. **Saeed, B.N. (2001)**. *Introduction to Robotics Analysis, Systems, Applications*. Pearson Education, Prentice Hall.
 27. **Chacón, M., Sandoval, R., & Vega, J. (2015)**. *Percepción Visual Aplicada a la Robótica*. Alfaomega.
 28. **ARTE (2019)**. *A Robotics Toolbox for Education*. [Online] Available: http://arvc.umh.es/arte/index_en.html.

Article received on 11/03/2019; accepted on 10/08/2020.
Corresponding author is David Sáenz Zamarrón.