



Hardware Article

Design of an affordable IoT open-source robot arm for online teaching of robotics courses during the pandemic contingency



Victor H. Benitez ^{a,*}, Rodrigo Symonds ^b, David E. Elguezabal ^b

^aDepartment of Industrial Engineering, Universidad de Sonora, Hermosillo 83000, Mexico

^bTecnologico de Monterrey, School of Engineering and Sciences, Blvd. Enrique Mazón López 965, Hermosillo 83000, Sonora, Mexico

ARTICLE INFO

Article history:

Received 15 July 2020

Received in revised form 6 November 2020

Accepted 9 November 2020

Keywords:

Robot kinematics

Teaching robotics

Virtual laboratory

Online teaching

Educational innovation

Higher education

ABSTRACT

This article explains the design and construction of an affordable, open-source robot arm for online teaching of robotics courses. The main goal of the proposed robotic prototype is to deal with the current situation of pandemic contingency, where students and instructors cannot attend laboratory facilities in person. The robotic system has four main components: an electromechanical robot arm structure, a control system, a Wi-Fi communications module, and a human-machine interface. The IoT (Internet of Things) robot arm can be used to demonstrate important robotics topics such as direct and inverse kinematics, which are shown by programming simple and complex motions using the Denavit-Hartenberg (DH) methodology. The capabilities of the robotic system are empowered by IoT technology, which is demonstrated with an HMI interface deployed in a smartphone using wireless Wi-Fi communication through an ESP32 microcontroller. The arm's purpose is to be a low-cost and replicable robot that aids the comprehension of robotics design through project-based learning, from the theoretical aspects to the actual coding and construction of a prototype.

© 2020 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Specifications table

Hardware name	Two-link Robot Arm
Subject area	● Educational Tools and Open-Source Alternatives to Existing Infrastructure
Hardware type	● Electrical engineering and computer science ● Mechanical engineering and materials science
Open Source License	GNU General Public License (GPL)
Cost of Hardware	\$28.48 USD
Source File Repository	https://doi.org/10.17605/OSF.IO/EN9W6

* Corresponding author.

E-mail addresses: vbenitez@industrial.uson.mx (V.H. Benitez), A00960460@exatec.tec.mx (R. Symonds), A00226458@exatec.tec.mx (D.E. Elguezabal).

<https://doi.org/10.1016/j.ohx.2020.e00158>

2468-0672/© 2020 The Author(s). Published by Elsevier Ltd.

This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Hardware in context

The COVID-19 has been declared a worldwide pandemic. This outbreak has had enormous consequences for educational institutions and raises questions regarding safety in laboratory facilities and the corresponding laboratory practices carried out there by students and research personnel. The situation is riskier or presents a higher risk of disease contagion for engineering students who perform experimental trials that require equipment and collaborative work. Most educational institutions have reduced the exposure by delivering virtual distance education to reduce the risk of contracting the virus. However, the experimentation required by laboratory practices is very difficult or impossible to achieve in a virtual environment. This has caused teaching to be seriously compromised when theory cannot be linked to practice. Some scholastic subjects that are practical or theoretical-practical have been suspended due to the confinement caused by the social distancing measures implemented by governments and health organizations to diminish the global prevalence of COVID-19.

The Robot Arm developed is a low – cost, robotic-arm-prototype designed to demonstrate the applications of direct and inverse kinematics using cheap and commercially available components. The motion control is achieved via an IoT interface integrated with low-cost and low-power microcontrollers. The system is validated using motion profiles with a vision-based tracking method. The two-link Robot Arm's main goal is to facilitate teaching under the pandemic conditions, supporting laboratory experiments that introduce important robotics topics, such as direct kinematics and inverse kinematics. The mechanical structure is shown in Fig. 1.

2. Hardware description.

The robot is a low-cost prototype which is powered by IoT components [1]. With this robotic system, inexperienced students, researchers, and even enthusiasts who do not have access to a robotics laboratory or a diversity of materials and actuators will be able to replicate, comprehend, and verify the resulting behavior motion of a robot built from scratch. Its links are made from PLA filament because of the low amount of material required to print each link and the flexibility and variety in design that a 3D printer allows. Also, because the amount of material used is minimal, the servo motors can lift the weight of the components used in the prototype.

This robot can be controlled by an HMI interface in a mobile phone to move the angle of each servo motor using Denavit - Hartenberg parameters to obtain the direct kinematics matrix and determine its position. Also, the interface can replicate the pattern of motion trajectories using inverse kinematics through geometrical relations between the links via a button that

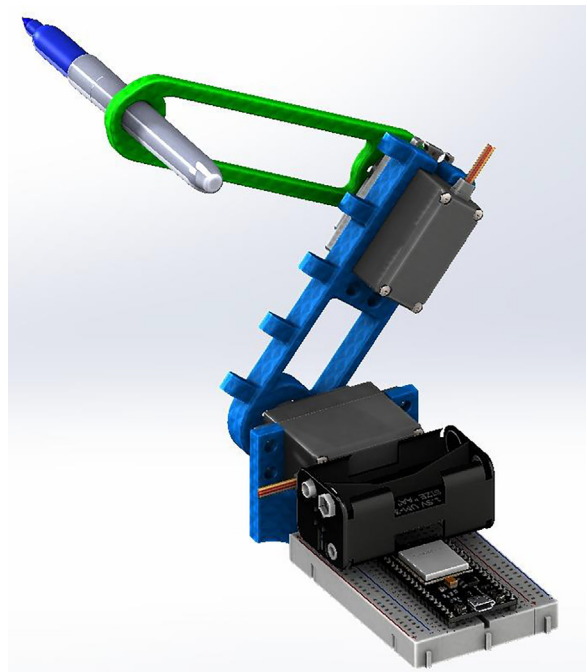


Fig. 1. CAD rendering of the robot in SolidWorks.

introduces a replication protocol that the robot follows. For purposes of demonstration, we selected the following patterns: a circle, a five-petal flower, and a spiral drawn by hand and processed with an open-source MATLAB code so that the robot could follow its trajectory. The equations of the robot were tested using a MATLAB that simulates the specified dimensions and trajectories the robot would follow in its physical prototype. To make the program accessible as an open-source project and fit the controller with wireless communication, we adapted the MATLAB code to be compatible with Arduino IDE. The program was loaded into ESP32 hardware because of its integrated Wi-Fi and dual-mode Bluetooth.

The researchers, teachers, or students who want to replicate the design would be able to:

- Comprehend the relationship between the equations and the behaviors of the robot.
- Implement their robotic configuration using this prototype as a guide, formulating the equations and executing the physical implementation.
- Be involved in the design of a physical prototype to demonstrate how knowledge is acquired through project-based learning.
- Obtain a low cost, functioning hardware for students to learn coding, electronics, and robotics.
- Use the robot to generate and control motion trajectories via an IoT interface in online experiments that simulate realistic environments.
- Take advantage of a highly customizable system that can be adapted to research and teaching needs.

3. Design files

Design Files Summary

Design file name	File type	Open source license	Location of the file
Robotic Link 1	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
Robotic Link 2	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
Servo Base	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
TOWER PRO MG-995, ANALOG SERVO	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://grabcad.com/library/tzt-mg-996r-servo-motor-1
Sharpie Fine Point	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://grabcad.com/library/sharpie-w-fine-point-1
BATTERY HOLDER	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://grabcad.com/library/battery-holder-aa-1-5v-1
ESP32S Dev Board 0.9 in. width ASSY	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://grabcad.com/library/esp-32s-development-board-1
proto_400	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://grabcad.com/library/protoboard-400-pts-1
Assem1	CAD File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
STL Base	STL File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
STL Link 1	STL File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
STL Link 2	STL File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6

(continued on next page)

a (continued)

Design file name	File type	Open source license	Location of the file
RobotArd	Arduino File	International License. Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
test1	PNG File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
test2	DAT File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6
digitize2	MATLAB File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://la.mathworks.com/matlabcentral/fileexchange/928-digitize2-m or available at https://doi.org/10.17605/OSF.IO/EN9W6
RobotFinalSIM	MATLAB File	Creative Commons Attribution-ShareAlike 4.0 International License.	https://doi.org/10.17605/OSF.IO/EN9W6

- **Robotic Link 1.sldprt** - SolidWorks part file of the first linkage in the robot. Consists of a hollow body with a slot for another servo and a circular servo horn.
- **Robotic Link 2.sldprt** - SolidWorks part file of the second linkage in the robot. Consists of a hollow body with a slot for a marker pen and a circular servo horn.
- **Servo Base.sldprt** - SolidWorks part file of the base of the robot that has a slot for the first servomotor and the 400-hole protoboard.
- **TOWER PRO MG-995, ANALOG SERVO.sldasm** - A SolidWorks assembly file found in grabcad.com, representing an MG-995 servo.
- **Sharpie Fine Point.sldprt** - A SolidWorks part file found in grabcad.com, representing a marker pen.
- **BATTERY HOLDER.step** - A SolidWorks part file found in grabcad.com, representing a 4-AA battery pack.
- **ESP32S Dev Board 0.9 in. width ASSY.sldprt** - A SolidWorks part file found in grabcad.com, representing an ESP32 development board microcontroller.
- **proto_400.step** - A SolidWorks part file found in grabcad.com, representing a small protoboard.
- **Assem1.sldasm** - A SolidWorks assembly file of the fully assembled two-link robot.
- **STL Base.stl** - A STL File of the robot's base for 3D printing.
- **STL Link 1.stl** - A STL File of the robot's first arm for 3D printing.
- **STL Link 2.stl** - A STL File of the robot's second arm for 3d printing.
- **RobotArd.ino** - Arduino code that is run in the Esp32.
- **test1.png** - Image of a scanned handwritten spiral to test the image processing.
- **test2.dat** - Generic data file to replicate the image processing test.
- **digitize2.m** - MATLAB file by Anil Prasad (2001) to digitize points based on an image.
- **RobotFinalSIM.m** - MATLAB file that uses Robotics Toolbox 9th Edition by Peter Corke (2015) to simulate the robot behavior.

4. Bill of materials

Component	Number	Cost per unit - currency	Total cost - currency	Source of materials	Material type
Tower Pro MG-995 Servo	2 pcs	\$4.99 USD	\$9.98 USD	https://www.amazon.com/-/es/KAILEDI-Motors-Walking-Control-Arduino/dp/B083DZ88R6/ref=sr_1_2?__mk_es_US=%C3%85M%C3%85C5%BD%C3%95C3%91&crd=1WF554RUM9TCC&dchild=1&keywords=servo+motor+mg995&qid=1591495062&prefix=servo+motors+mg99%2Caps%2C210&sr=8-2	Electronic
PLA 3D Printing Filament	4.163 g	\$24.99 USD	\$1.034 USD	https://www.amazon.com/-/es/SUNLU-Filament-Printer-Printers-LightGold/dp/B07X54WLHX/ref=sr_1_12?__mk_es_US=%C3%85M%C3%85C5%BD%C3%95C3%91&crd=JW4POYB4QP3Y&dchild=1&keywords=pla+filament&qid=1591588087&prefix=pla%2Caps%2C217&sr=8-12	Polymer
ESP32 Microcontroller	1 pc	\$10.99 USD	\$10.99 USD	https://www.amazon.com/HiLetgo-ESP-WROOM-32-Development-Microcontroller-Integrated/dp/B0718T232Z/ref=sr_1_1_sspa?__mk_es_US=%C3%85M%C3%85C5%BD%C3%95C3%91&dchild=1&keywords=esp32&qid=1591495666&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGkUXVhbGlmaWVyPUExUE1KN1Y0SU1TSU5HJmVuY3J5cHRIZElkPUExMDM1NzIxV1VWV1ZZREc5TkIvJmVuY3J5cHRIZEFkSWQ9QTAYMDg0MzEyTzNRU1ZWVVFjUQTFOjndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbWV1ZGlyZWNoJmRvTm90TG9nQ2xpY2s9dHJ1ZQ==	Electronic
Battery pack 4 AA	1 pc	\$5.99 USD	\$5.99 USD	https://www.amazon.com/Velleman-BH343B-Battery-AA-Cell-Terminals/dp/B00A41MZ62/ref=sr_1_37?dchild=1&keywords=4+aa+battery+pack&qid=1591565808&sr=8-37	Electronic
Jumper wires	10 pcs	\$5.79 USD	\$0.4825 USD	https://www.amazon.com/-/es/EDGELEC-Breadboard-Optional-Assorted-Multicolored/dp/B07GD2BWPY/ref=sr_1_3?__mk_es_US=%C3%85M%C3%85C5%BD%C3%95C3%91&dchild=1&keywords=jump+wires&qid=1591588133&sr=8-3	Electronic

5. Build instructions

The first step in building the robot is to acquire and print in 3D all the parts mentioned in the Bill of Materials and the design file summary. Once the parts are ready, the steps are the following:

- Insert the circular servo horns (included with the MG-995 servo) into the two robotic links and secure them with small screws.
- Secure the first servo motor into the servo base and the second servo motor into the first link slot with screws and nuts (6/32 in. screws work well).
- Secure the first robot link into the servo at the base by the circular horn and the second robot link at the second servo motor.
- Place the protoboard at the base of the robot and secure the ESP32.
- Connect the battery pack to the protoboard and connect the grounds of the servo and the battery pack.
- Finally, connect the first and second servo motors to ground, power, and the first and third pins of the microcontroller (Figure 2 and Figure 3).

5.1. Direct kinematics

The first step necessary to program the robot is to define its initial position and the workspace available with the servo motors. For convenience, this prototype assumes that both servos are at 0° as the following image shows, where S0 and S1 are the two locations of the servo motors, and S2 is the position at the end of the second link. Also, a_1 and a_2 are the lengths of each link, respectively.

It is also recommended to build a parameter table to visualize the previous configuration in a sectioned manner and comprehend the relationships between the actuators' behaviors.

- q_n : Angle input to the servo motor.
- θ_n : Desired angle for the robot.
- a_n : Length of arm or link.
- d_n : Offset between links (in this case, both are " because the robot is planar).
- α_n : Twist (used when a Z-axis is not aligned with the others in the D-H configuration).

Once the configuration is defined, the direct kinematics matrix can be obtained. Its general structure is as follows:

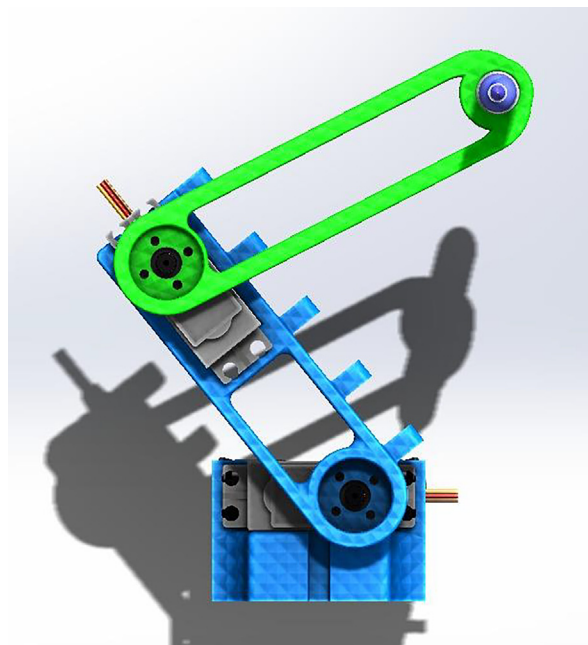


Fig. 2. Front view of the robot, showing how the arms are installed.

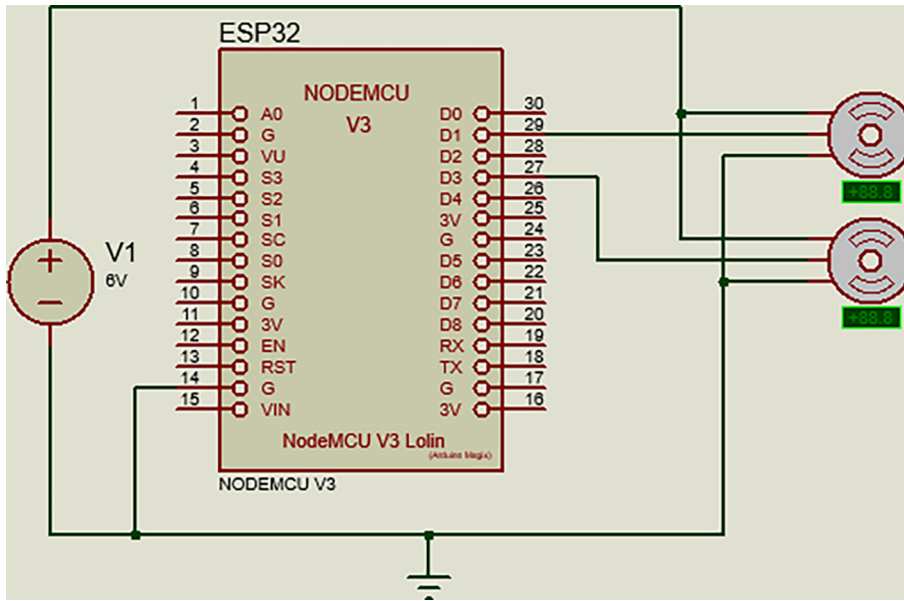


Fig. 3. Electrical diagram of the robot.

$$T_n = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n) \cos(\alpha_n) & \sin(\theta_n) \sin(\alpha_n) & a_n \cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n) \cos(\alpha_n) & -\cos(\theta_n) \sin(\alpha_n) & a_n \sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

So, by taking into consideration the values from Table 1 and Eq. (1), a matrix for each link is obtained.

$$T_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 84.8 \cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 84.8 \sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$T_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 106 \cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 106 \sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Multiplying (2) and (3) the following general direct kinematics matrix is obtained:

$$T = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 & 84.8 \cos(\theta_1) + 106 \cos(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 & 84.8 \sin(\theta_1) + 106 \sin(\theta_1 + \theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Table 1
D-H parameters from the chosen configuration.

Link	q _n	a _n	d _n	α _n
S1	q1 = θ1	a1 = 84.8 mm	d1 = 0	α1 = 0
S2	q2 = θ2	a2 = 106 mm	d2 = 0	α2 = 0

In the home position, the matrix (4) is evaluated numerically as

$$\text{where } \theta_1 = 0^\circ \text{ and } \theta_2 = 0^\circ \rightarrow T = \begin{bmatrix} 1 & 0 & 0 & 190.8 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In the previous matrix, the first three columns and rows correspond to the current axis of the final effector in relationship to the global axis of the system defined in the DH configuration shown in Fig. 4. For example, the number in the first row and column is 1, and so it indicates that the X-axis is in line with its original intended direction at 0° in both servo motors. The final column corresponds to the actual coordinate of the final effector. In the previous configuration, the effector is at (190.8, 0), or 190.8 mm to the right from the position (0, 0).

5.2. Inverse kinematics

To implement inverse kinematics by geometric relations correctly, one must be sure that the equations of the robot have all the necessary elements to find the desired angle, given a set of coordinates. There are multiple solutions to a single instruction, depending on how the equations are solved. To make a solution proposal, one must first put the robot in a position where all its angles are visible. The HOME position cannot be used because, in that position, the angles are not visible. Hence, the proposal is to begin with the inverse kinematics solution of this robot and to move to the second link, as depicted in Fig. 5:

where:

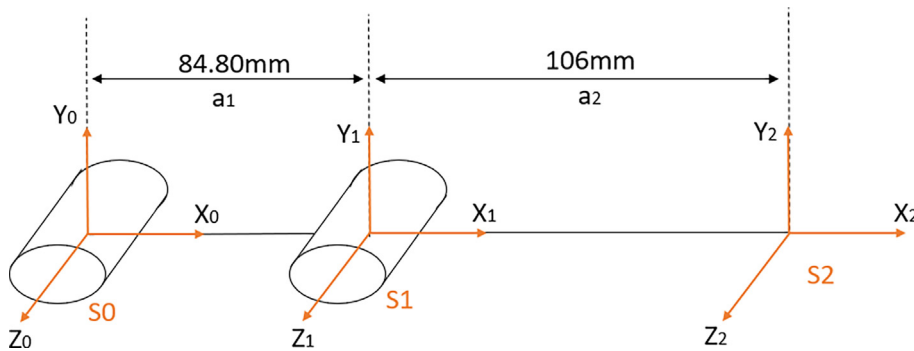


Fig. 4. Denavit-Hartenberg configuration for the robot in the HOME position.

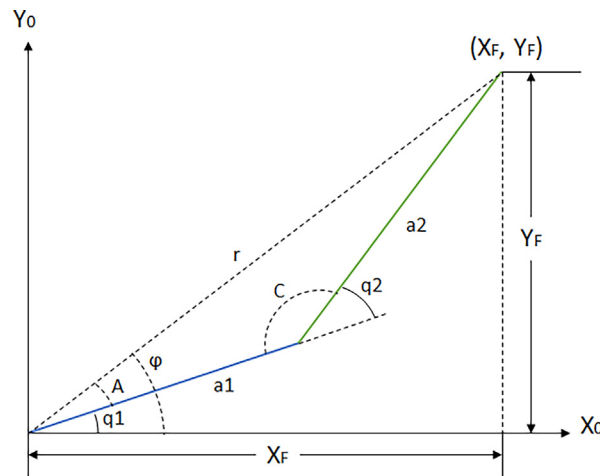


Fig. 5. Proposed robot position for inverse kinematic analysis.

- a1 and a2 are the lengths of each link.
- q1 and q2 are the desired angles the robot must determine.
- XF, YF, r, and φ are auxiliary variables for finding relations between the links.
- A and C are auxiliary variables to find relations between the links using trigonometric relationships.

After defining the model of Fig. 5, the equations of q1 and q2 can be found using trigonometric and algebraic knowledge. The Fig. 5 shows that it is easy to verify that:

$$r^2 = X_F^2 - Y_F^2 \quad (5)$$

Proposal to find q1:

$$q_1 = \varphi - A \quad (6)$$

$$\varphi = \tan^{-1}\left(\frac{Y_F}{X_F}\right) \quad (7)$$

Applying the Law of Cosines for auxiliary angle A:

$$a_2^2 = r^2 + a_1^2 - 2ra_1 \cos(A) \quad (8)$$

$$\cos(A) = \frac{r^2 + a_1^2 - a_2^2}{2ra_1} \quad (9)$$

Using trigonometric identity to find the sine of auxiliary angle A:

$$\sin(A) = \sqrt{1 - \left(\frac{r^2 + a_1^2 - a_2^2}{2ra_1}\right)^2} \quad (10)$$

Eqs. (9) and (10) are used to find the tangent of auxiliary angle A:

$$A = \tan^{-1}\left(\frac{\sqrt{1 - \left(\frac{r^2 + a_1^2 - a_2^2}{2ra_1}\right)^2}}{\frac{r^2 + a_1^2 - a_2^2}{2ra_1}}\right) \quad (11)$$

By substituting (7) and (11) into Eq. (6), it is possible to find the generalized coordinate q1 as:

$$q_1 = \tan^{-1}\left(\frac{Y_F}{X_F}\right) - \tan^{-1}\left(\frac{\sqrt{1 - \left(\frac{r^2 + a_1^2 - a_2^2}{2ra_1}\right)^2}}{\frac{r^2 + a_1^2 - a_2^2}{2ra_1}}\right) \quad (12)$$

From Fig. 5, it is possible to find q2:

$$q_2 = \pi - C \quad (13)$$

The Law of Cosines for auxiliary angle C:

$$r^2 = a_1^2 + a_2^2 - 2a_1a_2 \cos(C) \quad (14)$$

$$\cos(C) = \frac{a_1^2 + a_2^2 - r^2}{2a_1a_2} \quad (15)$$

Using trigonometric identity to find the sine of auxiliary angle C:

$$\sin(C) = \sqrt{1 - \left(\frac{a_1^2 + a_2^2 - r^2}{2a_1a_2}\right)^2} \quad (16)$$

Eqs. (15) and (16) are used to find the tangent of auxiliary angle C:

$$C = \tan^{-1}\left(\frac{\sqrt{1 - \left(\frac{a_1^2 + a_2^2 - r^2}{2a_1a_2}\right)^2}}{\frac{a_1^2 + a_2^2 - r^2}{2a_1a_2}}\right) \quad (17)$$

By substituting (17) into (13):

$$q_2 = \pi - \tan^{-1}\left(\frac{\sqrt{1 - \left(\frac{a_1^2 + a_2^2 - r^2}{2a_1a_2}\right)^2}}{\frac{a_1^2 + a_2^2 - r^2}{2a_1a_2}}\right) \quad (18)$$

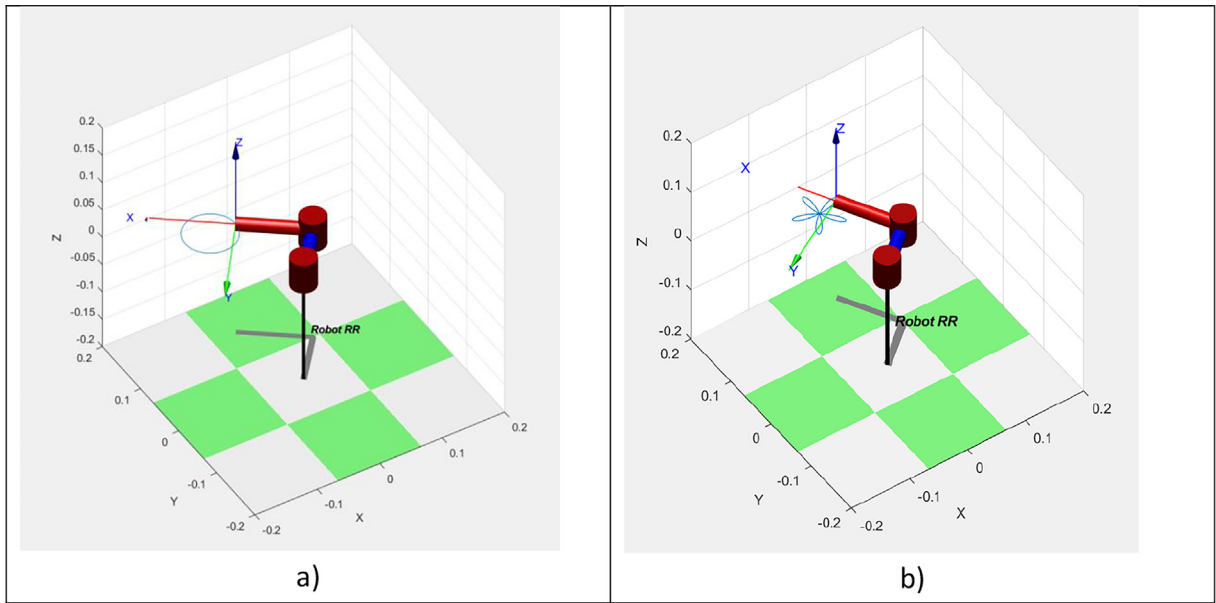


Fig. 6. Simulation of the robot replicating a circle, shown in subfigure a), and the five-petal flower, depicted in subfigure b), from parametric equations of inverse kinematics.

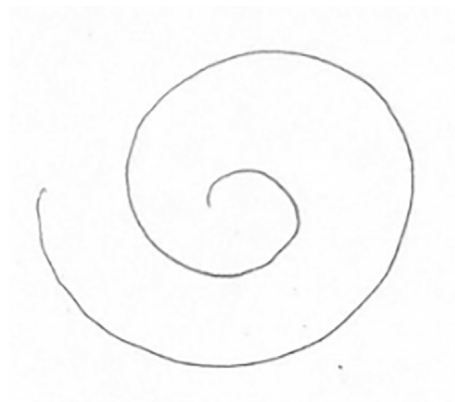


Fig. 7. Handwritten spiral scanned to be replicated by the robot.

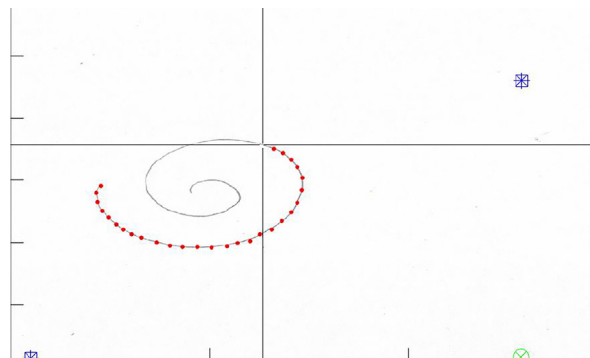


Fig. 8. Example of the process of point selection from an image.

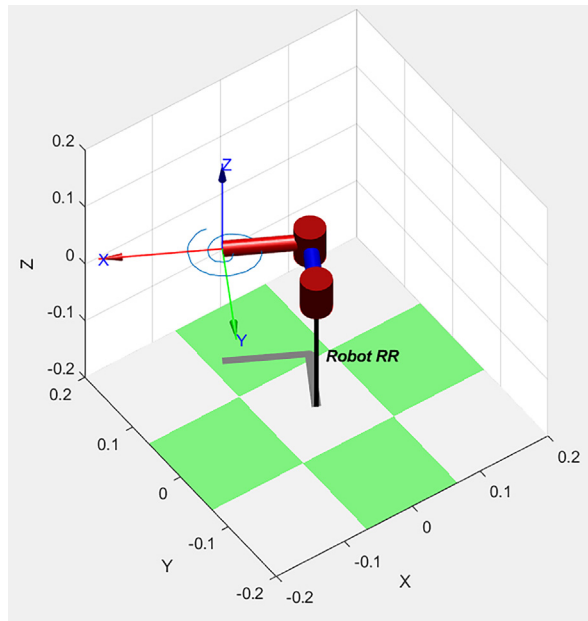


Fig. 9. Simulation of the robot replicating the handwritten spiral.

5.3. Simulations

The equations obtained above can be implemented directly in Arduino. However, to ensure that the robot behaves adequately, it is recommended that inverse kinematic Eqs. (12) and (18) are tested in simulation. A simulation code in MATLAB based on [2] was designed using Robotics Toolbox 9th Edition by Peter Corke (2015). The simulation is an additional step that safely tests the robot while implementing different parametric equations of determined shapes and observing that the obtained proposal for angles q_1 and q_2 works as intended. Because the simulation can replicate multiple shapes, the user must comment on all parametric equations except the one that wants to be tested. The MATLAB file to test this robot is included in the Source File Repository.

As Fig. 6a and Fig. 6b demonstrate, the robot indeed replicates the desired figures with the proposed solution obtained in (12) and (18). To further test the capabilities of the robot, we implemented a way to replicate any custom figure input by the

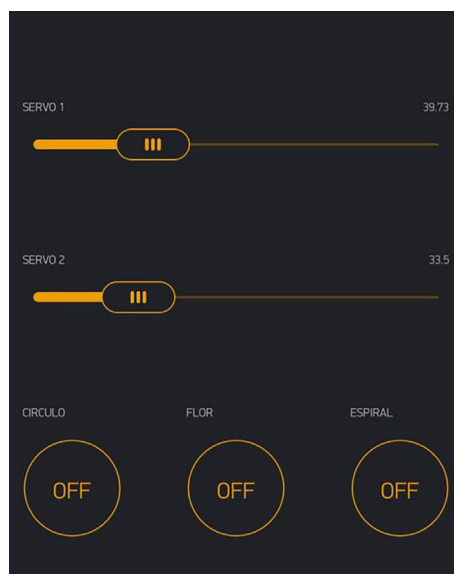


Fig. 10. Blynk smartphone app that is used to control the robot.

user. This method uses an open-source MATLAB program uploaded to the MATLAB Central File exchange [3]. It is a code that allows the user to digitalize a set of points from an image. For this test, a spiral was scribbled on a paper with no guidelines or symmetry, as shown in Fig. 7 and its processing output in Fig. 8.

The simulation output custom made spiral is depicted in Fig. 9.

5.4. IoT interface

To empower the capabilities of the robot, we specified that the robot should be capable of working online and being controlled by a smartphone. To achieve this, we decided to use the Blynk application [4]. Blynk is a smartphone app with com-

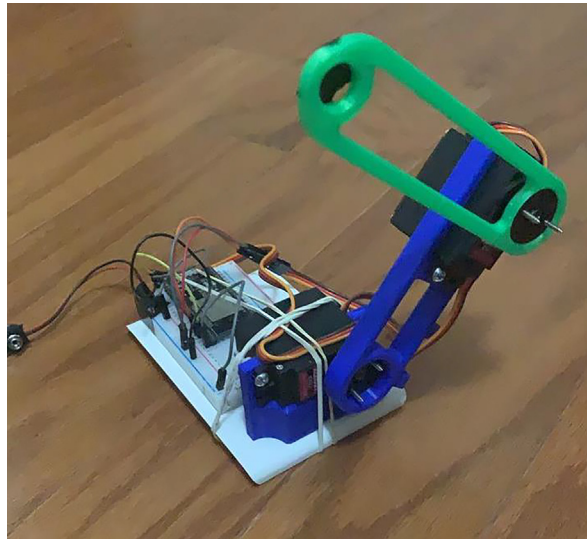


Fig. 11. Robot prototype (Front view).

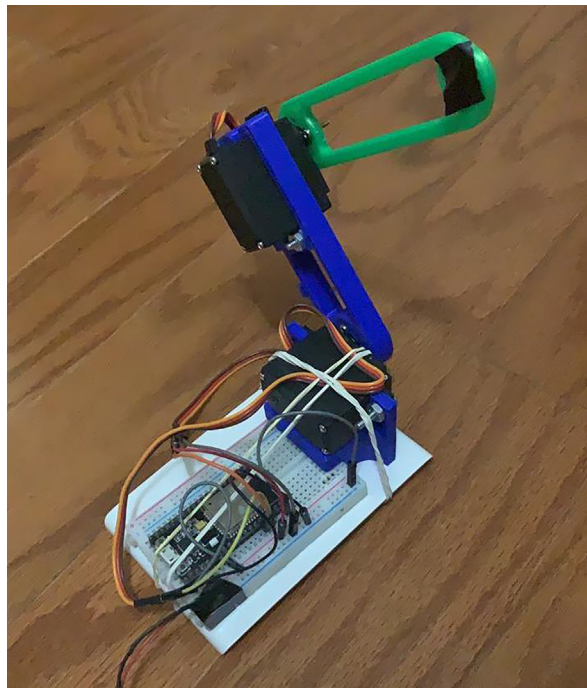


Fig. 12. Robot prototype (Back view).

plete documentation and easy-to-understand functions, allowing the user to create a server and a connection between a microcontroller and a smartphone via Arduino for free, as well as a completely customizable interface for each project. The documentation on the Blynk website allowed us to develop the interface and the connection with the microcontroller in a matter of hours; the only remaining problem was to translate the MATLAB code to Arduino.

The following was achieved with the math.h library in Arduino that adds some important functions for inverse kinematics, such as atan2(). The equations themselves, both being written in c, are similar in each program, making their translation an easy step. Finally, for each parametric equation or shape desired, a function was implemented that enclosed all the code needed for that specific movement, as well as the coordinates arrays used for the spiral shown above. The interface with the kinematic implementation is shown in the Fig. 10.

6. Operation instructions

- After adjusting the authentication token and the Wi-Fi connection in the Arduino code, connect the microcontroller to the computer via USB and upload the code.

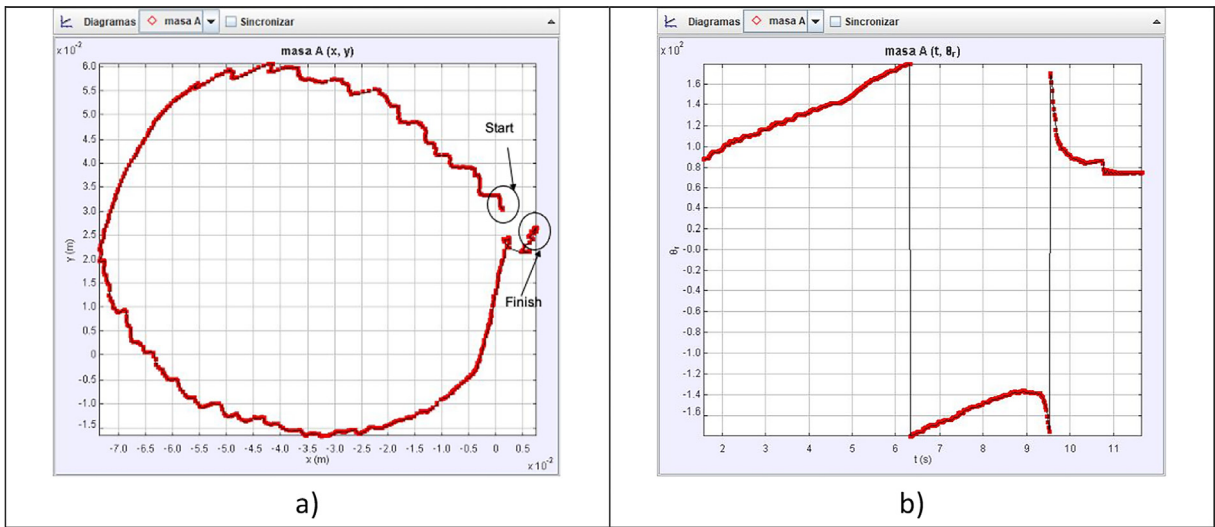


Fig. 13. Circle pattern tracked by the physical prototype, shown in subfigure a), and the angle vs time, depicted in subfigure b).

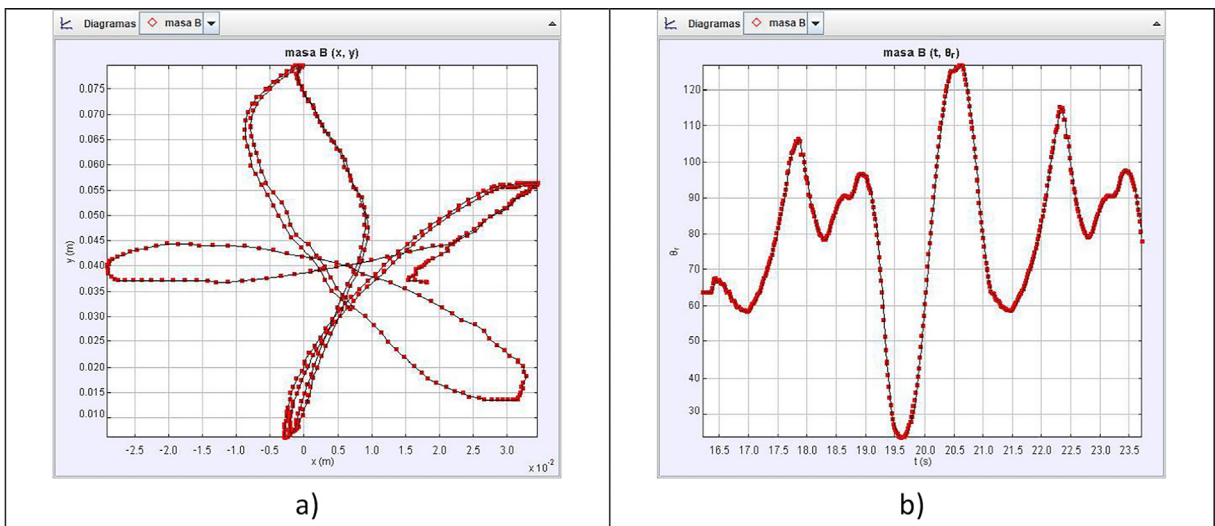


Fig. 14. Five-petal flower pattern tracked by the physical prototype, shown in subfigure a), and the angle vs time, depicted in subfigure b).

- While uploading code to the microcontroller, make sure that the power supply (6 V battery pack) for the servos is disconnected. While uploading, the pins on the ESP32 might send random signals and cause the servos to move if the battery is still connected.
- As soon as the code is uploaded or the coded ESP32 is connected for the first time, the ESP32 will attempt to connect to the internet.
- If the Wi-Fi signal is strong enough, the microcontroller will appear available in the Blynk app.
- From the phone, control the angle of each servo individually from 0 to 180° with the sliders and activate the pre-programmed functions with the buttons.
- Before changing the values of the rotation, the parametric functions, or the set of coordinates in the robot's code, make sure that the values you are attempting are real values that the robot can fulfill. When using wrong values, the servos can make abrupt movements that may damage the robot and their surroundings.

7. Validation and characterization

To demonstrate the operation and performance of the hardware, we built a prototype using the components described in the Bill of Materials (Figs. 11 and 12).

To evaluate the motion control of the prototype, we made a video of the robot replicating the patterns in Figs. 6 and 9 and loaded it into Tracker. Tracker is a free video-analysis-and-modeling tool built on the Open Source Physics (OSP) Java framework [5]. When a reference point to be followed is defined, the program can track the pattern that follows the established reference frame by frame. It compares the position and color of the pixels defined as references. The results of the physical prototype obtained when analyzing the trajectories were plotted automatically to see the pattern followed in the tests.

Parametric equation used to code the trajectories are:

Parametric curve used for the circle:

$$F_x[i] = -0.07 + 0.04 \cdot \cos(\text{rad});$$

$$F_y[i] = 0.13 + 0.04 \cdot \sin(\text{rad});$$

Parametric curve used for the flower:

$$F_x[i] = -0.06 + 0.04 \cdot (1 + \sin(5 \cdot \text{rad})) \cdot \cos(\text{rad});$$

$$F_y[i] = 0.1 + 0.04 \cdot (1 + \sin(5 \cdot \text{rad})) \cdot \sin(\text{rad});$$

Parametric curve used for the spiral:

As it was done using digitize library for MATLAB, the coordinates for the spiral are obtained as a data table, meaning that our program goes throughout different saved points plotting them in a discrete sequence.

As Figs. 13–15 show, the robot motion only approximates the real trajectories that the robot must follow. This is due to the following factors:

- The weight of the servos used represent extra effort not considered in Eqs. (12) and (18).
- The quality of the servo motors is not optimal but is affordable for the demonstration.
- The dynamic model of the robot was not considered.
- The friction between mechanical components.

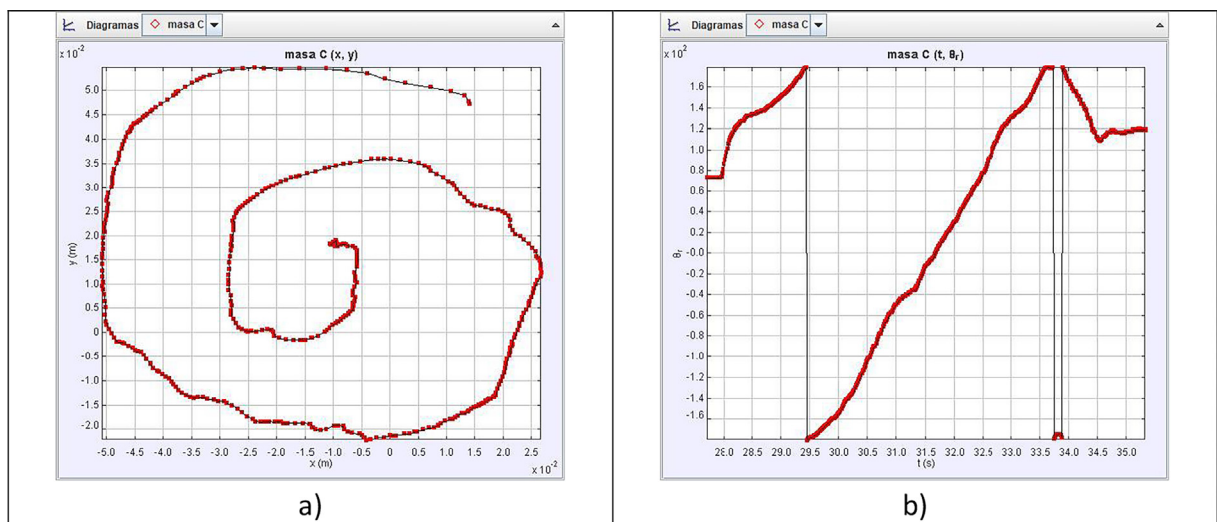


Fig. 15. Handwritten spiral pattern tracked by the physical prototype, shown in subfigure a), and the angle vs time, depicted in subfigure b).

- Arduino libraries accurate issues [6].
- Information is lost when transferring information to the servo motors using the PWM signal.

Nonetheless, the results obtained by the prototype are acceptable and adequately demonstrate the correct design and operation of a robot arm. A video clip is provided in the Source File Repository, where the motion tracking of patterns is shown.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper acknowledges Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Campus Sonora Norte (CSN), for the support in this work.

The authors acknowledge the technical support of Writing Lab, TecLabs, Tecnológico de Monterrey, Mexico, in the production of this work.

Human and animal rights

None.

Appendix A. Supplementary data

Supplementary data to this article can be found online at <https://doi.org/10.1016/j.ohx.2020.e00158>.

References

- [1] D. Paret, P. Crégo, Definitions and Position, in: D. Paret (Ed.), *Wearables, Smart Textiles and Smart Apparel*, Elsevier, 2019, 5-8.
- [2] P. Corke, *Robotics, Vision and Control, Fundamental Algorithms in MATLAB®*, Springer, Berlin, Heidelberg, 2011.
- [3] A. Prasad, digitize2.m. <https://la.mathworks.com/matlabcentral/fileexchange/928-digitize2-m>, 2001, (accessed 14.07.2020).
- [4] Blynk Inc., <https://blynk.io/>, 2020, (accessed 14.07.2020).
- [5] D. Brown, Tracker. Video Analysis and Modeling Tool, <https://physlets.org/tracker/>, 2002 (accessed 14.07.2020).
- [6] <https://www.arduino.cc/en/Reference/ServoWriteMicroseconds>, 2020, (accessed 23.09.2020).