Recibido: 15 de agosto de 2020
Aceptado:18 septiembre de 2021

INTEGRATION OF A COMMERCIAL COBOT WITH A CUSTOM-MADE LINEAR GUIDE AND AN ARTIFICIAL VISION SYSTEM FOR THE AUTOMATION OF A MANUFACTURING PROCESS

INTEGRACIÓN DE UN COBOT COMERCIAL CON UNA GUIA LINEAL HECHA A LA MEDIDA Y UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA AUTOMATIZACIÓN DE UN PROCESO DE MANUFACTURA

PhD. Sebastián Roa Prada [*]

[*] **Universidad Autónoma de Bucaramanga,** Facultad de Ingeniería, Programa de Ingeniería Mecatrónica.
Avenida 42 #48-11, Bucaramanga, Santander, Colombia.
+57 607 6436111.
E-mail: sroa@unab.edu.co.

Resumen: Este documento presenta el diseño, construcción y validación de un prototipo a escala para la automatización de una estación de trabajo de un proceso de manufactura. En el proyecto se usaron programas como SolidWorks para el diseño estructural, y código en Python o Matlab para la conexión con el brazo robótico comercial utilizado y funcionamiento en general. Se utilizaron algoritmos de visión artificial para reconocimiento de la pieza a manipular, así como una interfaz humano-máquina para el control del prototipo.

Palabras clave: Cobot, espacio de trabajo, visión artificial, manufactura.

Abstract: This document presents the design, construction and validation of a scale prototype for the automation of a workstation of a manufacturing process. In the project, programs such as SolidWorks were used for the structural design, and code in Python or Matlab for the connection with the commercial robotic arm used and operation in general. Artificial vision algorithms were used to recognize the part to be manipulated, as well as a human-machine interface to control the prototype.

Keywords: Cobot, workspace, computer vision, manufacturing.

1. INTRODUCTION

This work proposes a solution to the need to reduce labor risks due to the work of operators in areas of high risk to health that can occur in different stages of manufacturing processes such as heat treatments in series production lines. This type of solution also allows us to think about the possibility of increasing production using a cobot, since, according to research carried out in 2016 by MIT, collaboration between humans and robots is 85% more productive than that of a person or robot working separately (Pelegrí J.).

According to statistics provided by the Ministry of Health, in 2018 in Colombia there were 527,859 qualified work accidents, and 10,410 diseases classified as occupational, of which 103,560 accidents and 3,047 diseases occurred in the manufacturing industry, presenting a rate of 7.18

qualified accidents per 100 affiliates and 274.25 diseases per 100,000 affiliates (Ministerio de Salud y de la Protección Social: Resumen estadisticas Sistema General, 2019).

Due to these drawbacks, the solution that was thought was to use a commercial collaborative robot to automate this process (Universal Robots, 2016), thus removing the current operator from the heat treatment station due to its high temperatures. The first collaborative robots were installed in industrial environments almost ten years ago, but they are quite unknown to the general public. In turn, the number of robots installed in factories has been increasing, and cobots are presented as the best option for industrial automation, being the key piece for the development of Industry 4.0. Sometimes, the tasks that a cobot is responsible for can be tasks that endanger the company's personnel. For example, in the UK, between 2015 and 2016, it is estimated that around 8 million working days have been lost due to work-related bone or muscle injuries. For these reasons, cobots are increasingly used to perform tasks in sectors such as construction, the automotive industry, driving or even health (Pelegrí J. , 2019).

Taking into account the above, this project is focused on developing a scale prototype that positions a piece manufactured in three stations where different thermal and machining processes will be carried out, this in order that the operator does not have to perform this task, which, in addition to being repetitive, is risky due to the high working temperatures, preventing possible health problems, in addition to making the work more efficient.

2. WORKSPACE

The first step before thinking about some possible solutions is to understand where the work is carried out, along with the dimensional constraints and limitations that it has. The heat treatment station for the pieces has 3 main zones: collection or starting station, two stations where each one performs a tempering and tempering to the piece, and the final station in which the piece is deposited, repeating this process with the other pieces that arrive at the workspace. The CAD model of the workspace, based on the measurements taken, is presented in Fig. 1.

After carefully observing the workstation and the characteristics of the UR3 robot acquired by the university, it was observed that it was necessary to build a linear guide that would give the robot a degree of freedom more and thus be able to meet the objective set.

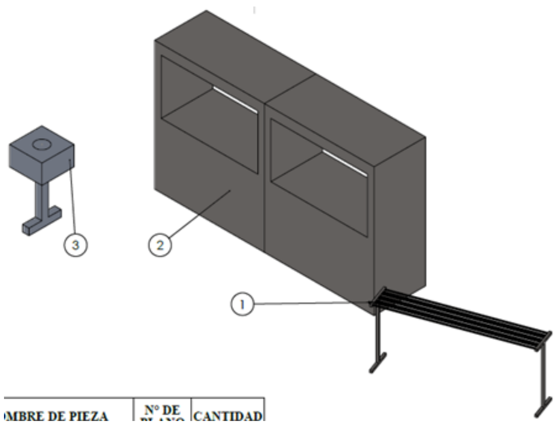


Fig 1. CAD Model of the workspace

3. CONSTRUCTION OF THE SEVENTH DEGREE OF FREEDOM

3.1 Design of the linear guide

Initially for the development of the linear guide in which the robot was positioned, the structural dimensions of the base were taken into account, in which the robot is located at the Autonomous University of Bucaramanga, already obtained this as a starting point we proceed to define what is the effective career that the linear guide must have to meet the requirements, this length was defined from the dimensions of the jobs and the scale at which this project was carried out. Taking all the above considerations into account, it was obtained that the effective stroke of the linear guide should be 1 meter.

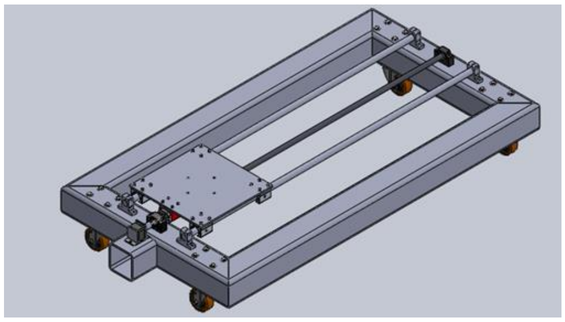


Fig 2. Seventh Degree of Freedom CAD Design

3.2 Prototype assembly

To build the base of the prototype, a 100mm x 100mm square tube was used, to which the necessary cuts were made and joined by SMAW welding. Once these joints were obtained, the protrusions produced by the weld were polished and annexed to this the holes in the structure in which the screws that will fix the wheels will go were drilled.

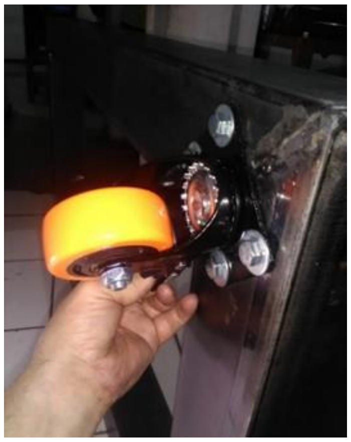


Fig 3. Linear guide wheels

After this, the anticorrosive and gray paint were applied.

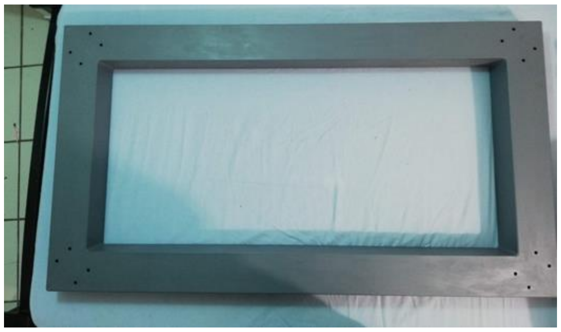


Fig 4. Structure of the linear guide

Once the paint was dry, the assembly of the smooth shafts was carried out which are made of 1020 steel of 20mm in diameter, these were assembled with SK20 supports and SC20 bearings, already made the assembly proceeds to buy a plate and the points in which the linear bearings will be fastened are marked and perforated, the nut base of the recirculating ball screw and the base of the UR3 robot, once the perforations have been made, the plate is polished and painted. Following this, the assembly of the recirculating ball screw is made, which is positioned in the middle of the smooth axes, it should be clarified that this was also alienated with the smooth axes (Norton, 2019).

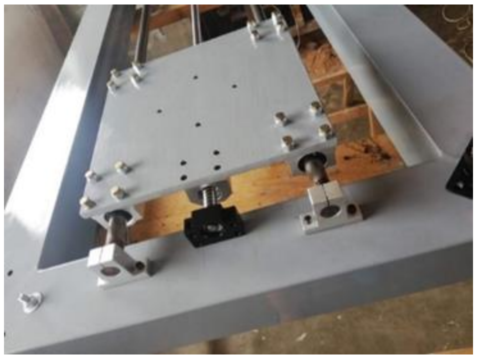


Fig 5. Construction of the guide

Following this, the stepper motor is assembled to the metal base and coupled to the recirculating ball screw.

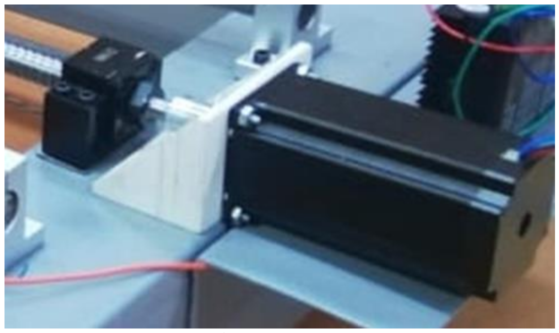


Fig 6. Assembly of the NEMA 23 motor to the structure

Once the mechanical assembly has been completed, the assembly of the electrical parts such as the end of the stroke, the connection terminal and the step-by-step motor controller is carried out.

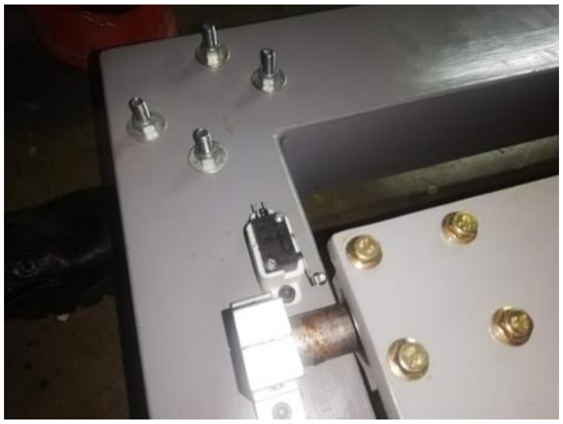


Fig 7. Base of the final stroke sensors

Once this was finished, the positioning of the terminal was carried out and the controller which was fixed on a metal support, being fixed in the assembly the wiring is carried out, this was done according to the plan shown in the following figure.

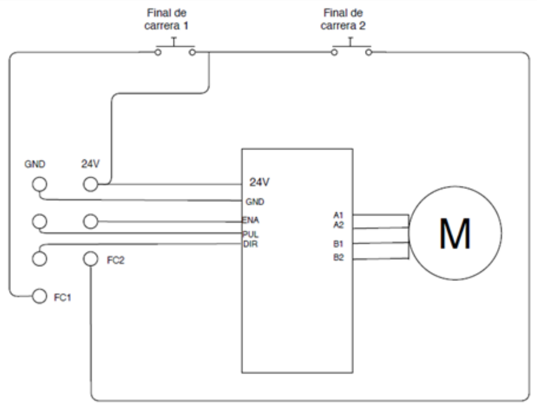


Fig 8. Wiring plan

Below, it is shown how the linear guide with its different components is built in its entirety.

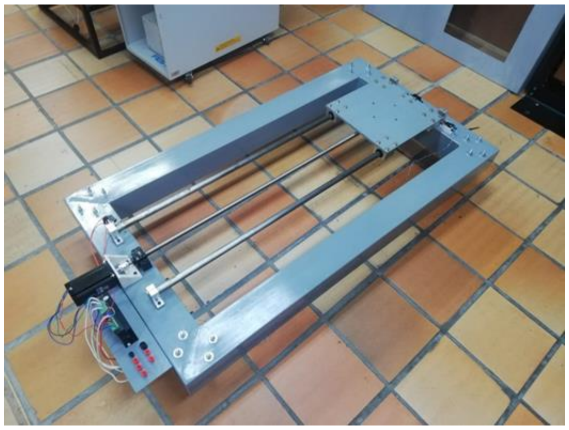


Fig 9. Fully built linear guide

4.MACHINE VISION SYSTEM

4.1 Machine vision methods

In the workspace under study, the two heat treatment stations and the final station are fixed, this implies that 3 of the 4 points of the route are completely known. However, the initial station is not attached to the ground, so it is possible to move it to different places, affecting the position of the piece at the collection site.

It is for this reason that an artificial vision system (Cuevas Jimenez, Zaldivar Navarro, Perez Cisneros, & Garcia Tome, 2010) (Lozano Mantilla & Orduz Rodriguez, 2015) was implemented, with the aim of identifying the position of the piece and using this information to indicate to the robot what will be the point of passage at which it should pick it up. An investigation was carried out of different methods with which this recognition could be carried out, in addition to carrying out tests with these.

4.2 Detect Surf Features

This process is based on using a reference image to detect the "Surf characteristics" of the object (see figure 10), and then compare them with the characteristics of other images, and by checking if matches are found, you can get a result from the presence or absence of the piece. (Bay, Tuytelaars, & Van Gool, 2006)

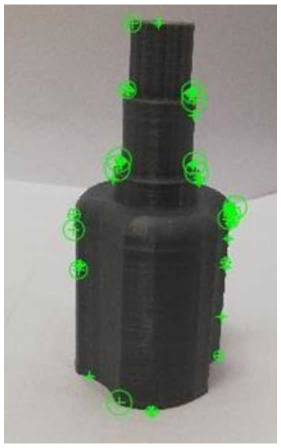


Fig 10. Surf characteristics of the piece to be handled

The advantage of this process is its robustness, that is, you can find these characteristics even when the object is rotated, scaled or partially hidden in the verification images, and have enough matches with the reference image to find the location within the verification photos.
By performing different tests in which the piece was surrounded by other objects, characteristics similar to those seen below were obtained:

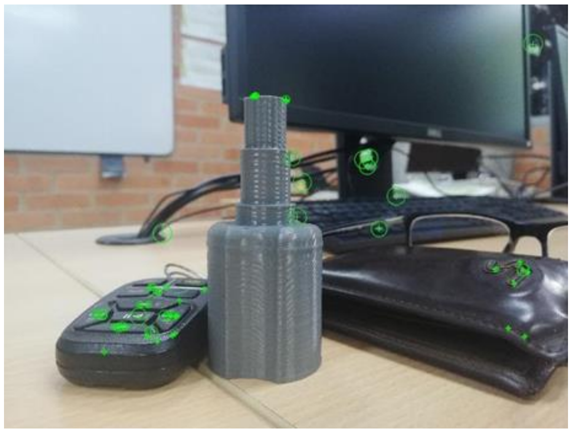


Fig 11. Surf features found in the validation photo.

It can be noted that many features were found that do not correspond to the part, and when comparing the characteristics to find matches, the results obtained by this method were not adequate:

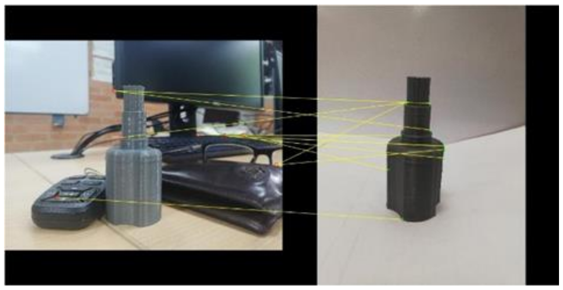


Fig 12. Matches found in the two images

For this point, tests were carried out obtaining the characteristics in different reference photos, but in none of the cases the coincidences obtained corresponded to the piece, so this method was discarded.

4.3 Pre-trained model

This method consists of taking a considerably large number of photos, with the aim of being processed to obtain the characteristics of the object that you want to recognize through what is known as a training. This was done with the YOLO (You Only Look Once) algorithm, which is an open-source system that makes use of a single neural network to detect objects in images (Redmon, Divvala, Girshick, & Farhadi, 2016). This is an iterative process, in which the weights of the features found are modified and saved in files called "models", which can be called in the main recognition code.

These identification photos must be manually tagged to indicate to the training code where the desired part is located:

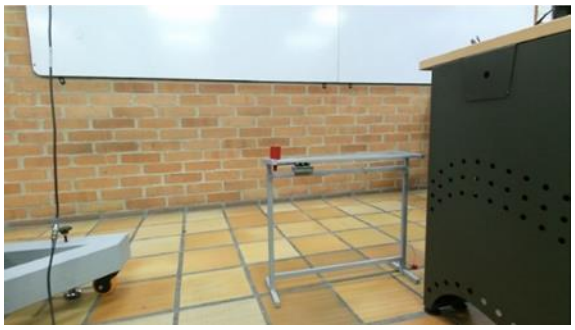


Fig 13. Example photo ID

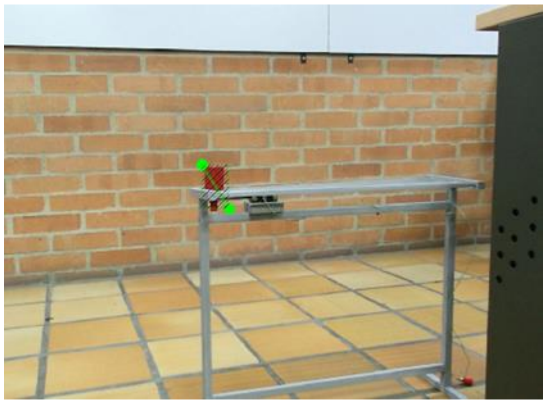
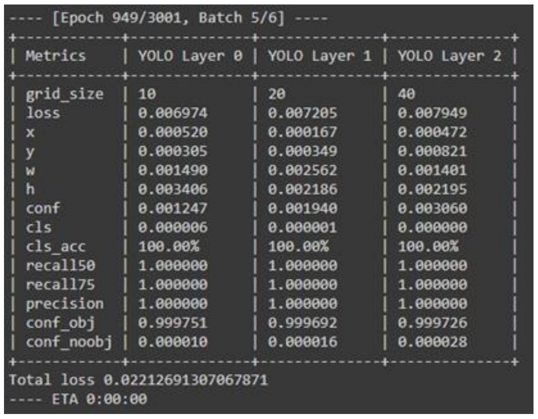


Fig 14. Manual labeling

In this way, after labeling each identification image, the training code is executed. In this case, about 950 iterations were carried out, from which information similar to that presented below can be obtained.


```
---- [Epoch 949/3001, Batch 5/6] ----
+------------+--------------+--------------+--------------+
| Metrics    | YOLO Layer 0 | YOLO Layer 1 | YOLO Layer 2 |
+------------+--------------+--------------+--------------+
| grid_size  | 10           | 20           | 40           |
| loss       | 0.006974     | 0.007205     | 0.007949     |
| x          | 0.000520     | 0.000167     | 0.000472     |
| y          | 0.000305     | 0.000349     | 0.000821     |
| w          | 0.001490     | 0.002562     | 0.001401     |
| h          | 0.003406     | 0.002186     | 0.002195     |
| conf       | 0.001247     | 0.001940     | 0.003060     |
| cls        | 0.000006     | 0.000001     | 0.000000     |
| cls_acc    | 100.00%      | 100.00%      | 100.00%      |
| recall50   | 1.000000     | 1.000000     | 1.000000     |
| recall75   | 1.000000     | 1.000000     | 1.000000     |
| precision  | 1.000000     | 1.000000     | 1.000000     |
| conf_obj   | 0.999751     | 0.999692     | 0.999726     |
| conf_noobj | 0.000010     | 0.000016     | 0.000028     |
+------------+--------------+--------------+--------------+
Total loss 0.02212691307067871
---- ETA 0:00:00
```
Fig 15. Training results in iteration number 949

One criterion for stopping training is to look at the value of total losses: when these are at values less than 1, it is an appropriate time to stop training. In this case, it can be noted that for this point the aforementioned criterion is met, so it was not necessary to continue training the model, which could lead to overtraining it.

It is not necessary to use the most current model to perform the recognition, since there are several parameters that can be modified in the code, which can drastically change the results. In this case, the model number 500 was used, which showed good performance. Using the image observed in Figure 13, the result shown in Figures 16 and 17 was obtained:

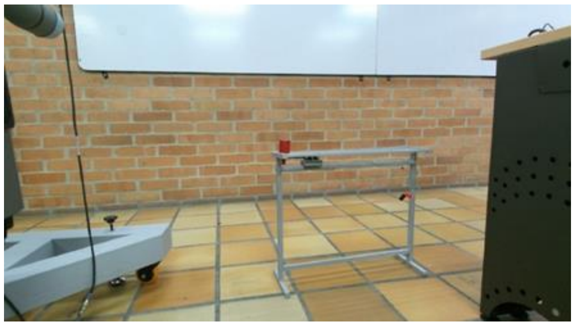


Fig 16. Model verification photo

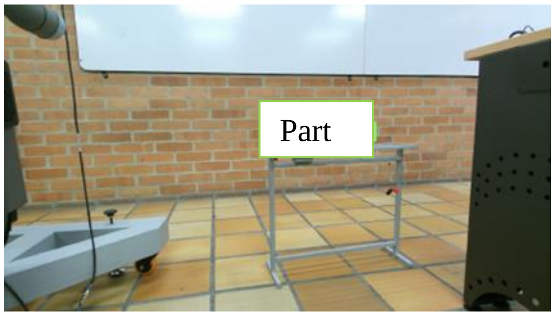


Fig 17. Result of the identification code using the model 500

NOTE: To verify proper operation, this process was repeated with different photos, which are all different from id photos.

Looking at the results of this method, it was chosen to be used in performance tests.

5.INTERFACE DEVELOPMENT

5.1 Selection of components for the interface

For the development of this interface, it was decided that the computer screen would be used, that is, no physical element (such as pushbuttons or switches) would be used. In this way, the number of connections is kept to a minimum.

However, as in any automated process, the operator must be given options to interact with the prototype: the power, pause and emergency stop buttons. In addition, in this specific case, it is important to keep the robot and the position of the part monitored, so in the interface it was decided to show these two characteristics, with an image of the position of the piece and the text fields corresponding to the position and orientation of the TCP and the links of the robot.

5.2 Programming the HMI interface

Once you have decided the elements that will be displayed in the interface, you must choose the programming language in which it will be made. The choice of language was made quickly, considering that the training and detection code were developed using Python, this was the logical option to develop the interface.

The Tkinter library was used, which provides all the necessary functions: elements such as buttons, labels, text fields, events, reading image files, etc ... To correctly track the robot process. For this, the interface must be connected to the robot, that is, it must send and receive data from it, the robot being especially attentive to the emergency stop button. Below, the result of the interface with the above mentioned is observed:

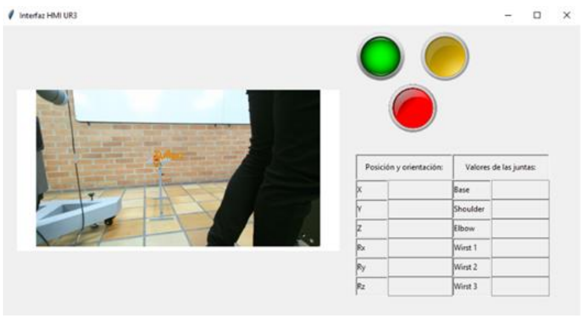


Fig 18. Final result of the interface

6. MODEL OF THE WORKSPACE

Initially the measurements between workstations were taken into account, for the development of this work it should be clarified that there are 4 stations in which the robot must position or remove the piece. Then, the workstation was made, for the development of this a design was made in SolidWorks and scaled, to obtain the exact measurements of the model, then the CAD design and the built model of the first 2 stations will be shown.

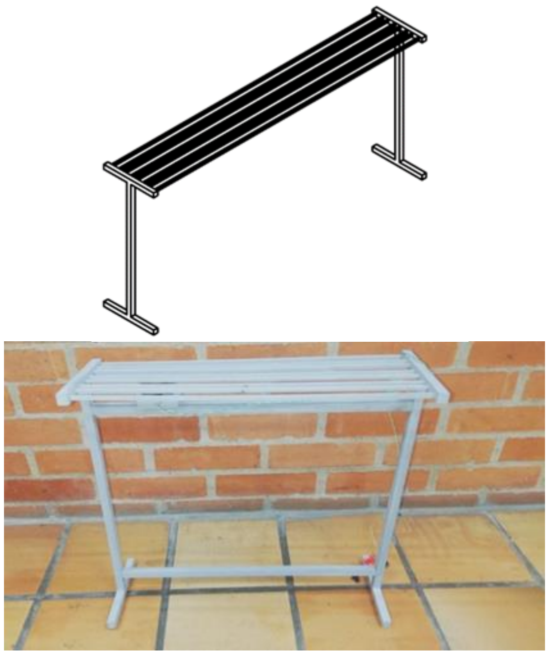


Fig 19. Workstation 1

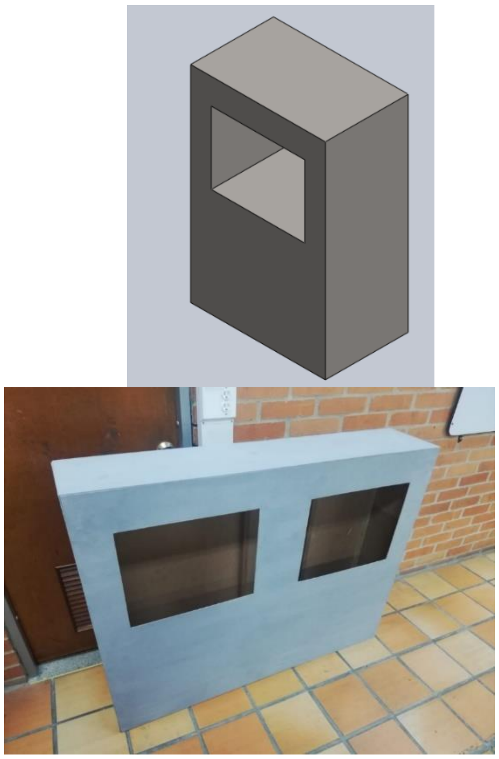


Fig 20. Workstation 2

And finally, the design and manufacture of the inspection station was developed in which the robot will position the part and end a work cycle, for this a CAD design and its scaling was also developed, and then it will be shown how the inspection station is in the model.

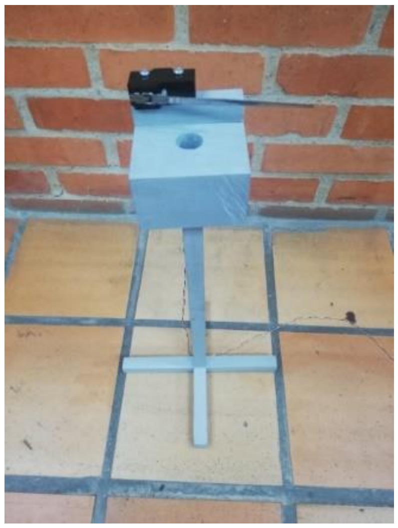


Fig 21. Workstation 3

Once the design of the model is finished, a scale design of the piece to be transported is carried out to carry out the relevant tests and the verification of the operation, for the design of this the approximate measurements were taken into account and by means of 3D printing, the construction of this was carried out.

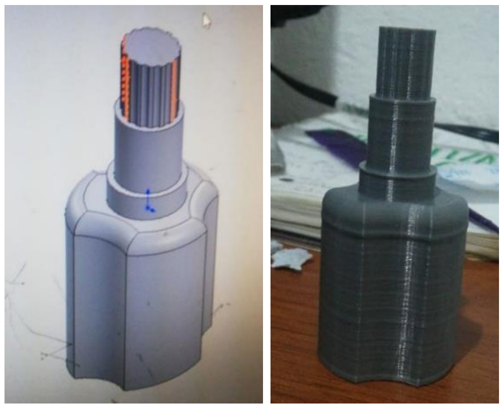


Fig 22. Part for transport

7. RESULTS

Up to this point, each of the sections of the project has been tested individually: the recognition of the piece implementing artificial vision, the movement and stop of the linear guide thanks to the reading of the sensors, and the movement of the robot's passage points.
However, all these processes must be executed at the right times automatically so that the robot's behavior is adequate. For this, it should be clarified that the connection between the robot and the computer was made using an ethernet cable, using

the TCP protocol to perform the exchange of information.

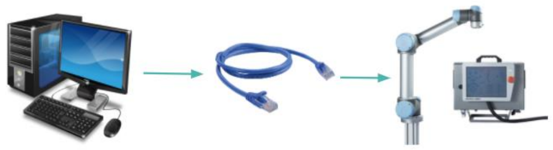
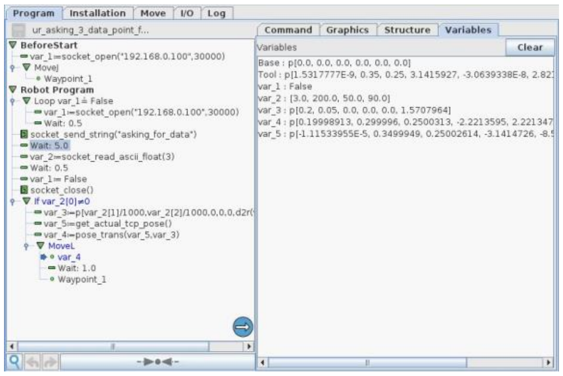


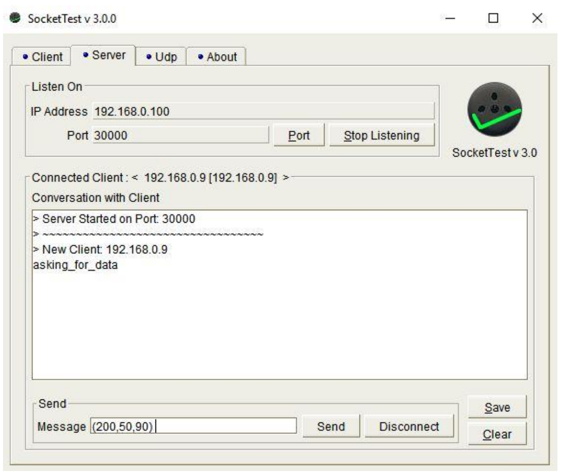
Fig 23. Connection between the computer and the cobot

As the connections were made, tests were also performed that they were working properly. For example, the sending and receiving of data between the robot and the computer was verified using the SocketTest program. The communication was initialized, and the program was used to send the desired location to the robot, and it read the data sent, processed it and converted it into a type of data "position", as seen in figures 24 and 25.



```
BeforeStart
  var_1≔socket_open("192.168.0.100",30000)
  MoveJ
    Waypoint_1
Robot Program
  Loop var_1≟ False
    var_1≔socket_open("192.168.0.100",30000)
    Wait: 0.5
  socket_send_string("asking_for_data")
  Wait: 5.0
  var_2≔socket_read_ascii_float(3)
  Wait: 0.5
  var_1≔ False
  socket_close()
  If var_2[0]≠0
    var_3≔p[var_2[1]/1000,var_2[2]/1000,0,0,0,d2r(
    var_5≔get_actual_tcp_pose()
    var_4≔pose_trans(var_5,var_3)
    MoveL
      var_4
      Wait: 1.0
      Waypoint_1
```

```
Variables
Base : p[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Tool : p[1.5317777E-9, 0.35, 0.25, 3.1415927, -3.0639338E-8, 2.82
var_1 : False
var_2 : [3.0, 200.0, 50.0, 90.0]
var_3 : p[0.2, 0.05, 0.0, 0.0, 0.0, 1.5707964]
var_4 : p[0.19998913, 0.299996, 0.2500313, -2.2213595, 2.221347
var_5 : p[-1.11533955E-5, 0.3499949, 0.25002614, -3.1414726, -8.5
```
Fig 24. Communication test script on the robot



IP Address 192.168.0.100
Port 30000
Connected Client : < 192.168.0.9 [192.168.0.9] >

```
> Server Started on Port: 30000
> ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> New Client: 192.168.0.9
asking_for_data
```

Message (200,50,90)

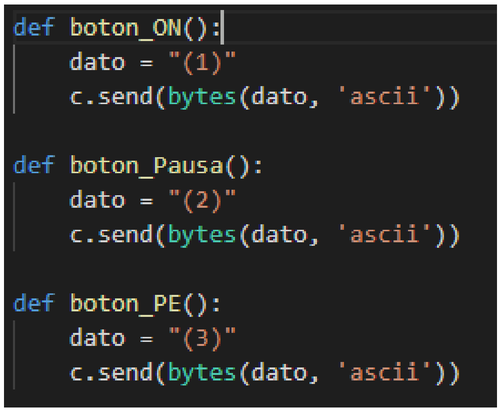
Fig 25. Communication test in SocketTest

However, this communication test cannot be carried out in the development of the project, since there is no direct way to communicate the program already developed in Python with this new software. This is why a library called Socket for Python was used, which allows you to use these same functions of sending and receiving information, also allowing you to adapt the use of this data to the needs of the project.

One of the problems that arose in the testing stage was how to read and differentiate the data that arrived to indicate a position, and those that referred to the start, pause and emergency stop commands, since, in case of sending data at different times than those established in the robot script, this would not read them or keep them on a waiting list to be read at the next reading time.

After several attempts and search in the reference manuals of the cobot, he came across the "subtask" section (see figure 27), which is a program that runs parallel to the main program, in which it is possible to execute data reading, modification of variables and control of other machines connected to the robot controller.
In turn, taking advantage of the feature of being able to specify the amount of data expected, a distinction can be made between the data referring to modification of the position of the robot, and the data that will be used to activate the functions of the buttons of the interface, since these have a different amount of data: while 6 data are needed to modify the position and orientation of the robot, with a single piece of data we can assign the desired behaviors of the buttons.

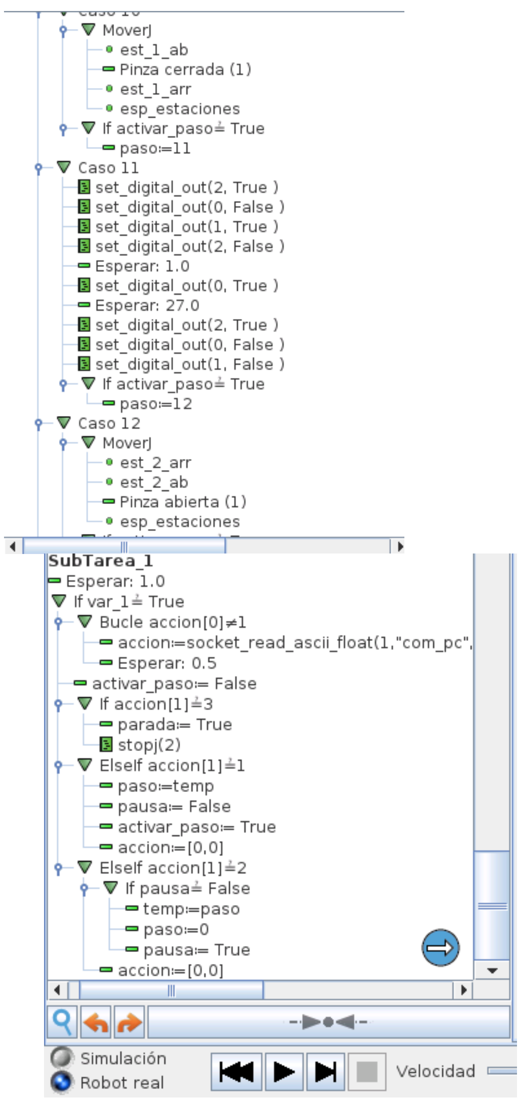
In this way, by sending a single data through the communications port, it is much easier to perform the desired action on the robot.



```python
def boton_ON():
    dato = "(1)"
    c.send(bytes(dato, 'ascii'))

def boton_Pausa():
    dato = "(2)"
    c.send(bytes(dato, 'ascii'))

def boton_PE():
    dato = "(3)"
    c.send(bytes(dato, 'ascii'))
```
Fig 26. Data sent according to the button pressed



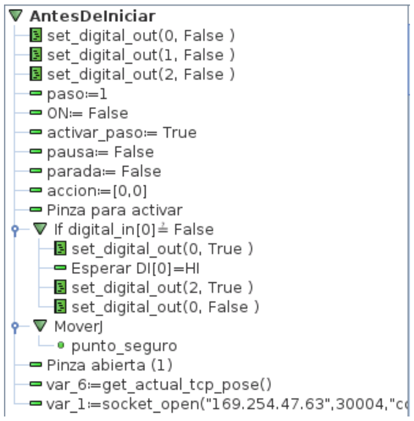
```
    MoverJ
      est_1_ab
      Pinza cerrada (1)
      est_1_arr
      esp_estaciones
    If activar_paso≟ True
      paso≔11
  Caso 11
    set_digital_out(2, True )
    set_digital_out(0, False )
    set_digital_out(1, True )
    set_digital_out(2, False )
    Esperar: 1.0
    set_digital_out(0, True )
    Esperar: 27.0
    set_digital_out(2, True )
    set_digital_out(0, False )
    set_digital_out(1, False )
    If activar_paso≟ True
      paso≔12
  Caso 12
    MoverJ
      est_2_arr
      est_2_ab
      Pinza abierta (1)
      esp_estaciones
```

```
SubTarea_1
Esperar: 1.0
If var_1≟ True
  Bucle accion[0]≠1
    accion≔socket_read_ascii_float(1,"com_pc",
    Esperar: 0.5
  activar_paso≔ False
  If accion[1]≟3
    parada≔ True
    stopj(2)
  ElseIf accion[1]≟1
    paso≔temp
    pausa≔ False
    activar_paso≔ True
    accion≔[0,0]
  ElseIf accion[1]≟2
    If pausa≟ False
      temp≔paso
      paso≔0
      pausa≔ True
    accion≔[0,0]
```

Fig 27. Subtask code

Finally, some changes to the program made earlier in the cobot movement tests were necessary:

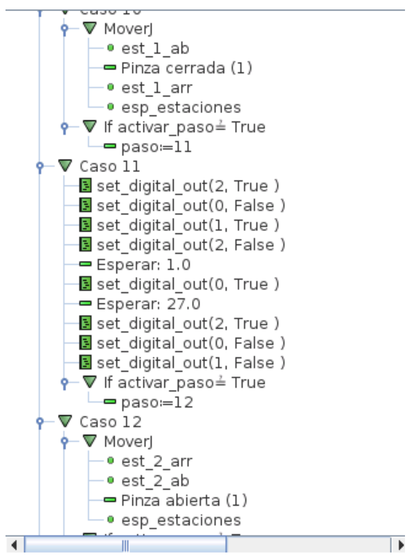
Initialization conditions seen in Figure 28, to start from the same point at the beginning of the process or after an emergency stop. This is done by activating the "Before Starting" section, included in the robot's programming options.



```
AntesDeIniciar
  set_digital_out(0, False )
  set_digital_out(1, False )
  set_digital_out(2, False )
  paso≔1
  ON≔ False
  activar_paso≔ True
  pausa≔ False
  parada≔ False
  accion≔[0,0]
  Pinza para activar
  If digital_in[0]≟ False
    set_digital_out(0, True )
    Esperar DI[0]=HI
    set_digital_out(2, True )
    set_digital_out(0, False )
  MoverJ
    punto_seguro
  Pinza abierta (1)
  var_6≔get_actual_tcp_pose()
  var_1≔socket_open("169.254.47.63",30004,"c
```

Fig 28. Initialization code on the robot

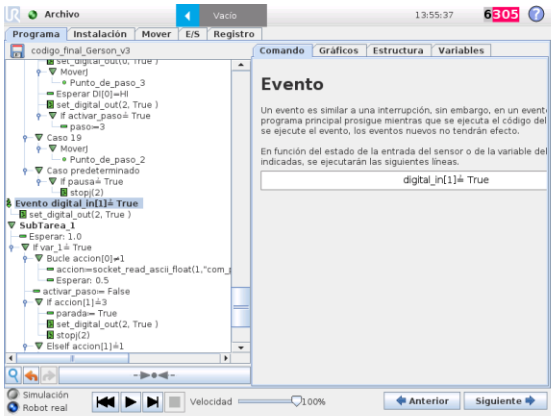
Add more intermediate cases for linear guide movements (such as those seen in Figure 29). This is because the tests carried out previously to check the operation of the interface motors were not carried out with the base in motion. However, it is really easy to add these to the sequence of the program, having only to add the necessary ones and the lines of script to write into the outputs and read the inputs of the robot.



```
    MoverJ
      est_1_ab
      Pinza cerrada (1)
      est_1_arr
      esp_estaciones
    If activar_paso≟ True
      paso≔11
  Caso 11
    set_digital_out(2, True )
    set_digital_out(0, False )
    set_digital_out(1, True )
    set_digital_out(2, False )
    Esperar: 1.0
    set_digital_out(0, True )
    Esperar: 27.0
    set_digital_out(2, True )
    set_digital_out(0, False )
    set_digital_out(1, False )
    If activar_paso≟ True
      paso≔12
  Caso 12
    MoverJ
      est_2_arr
      est_2_ab
      Pinza abierta (1)
      esp_estaciones
```

Fig 29. One of the code sections added for guide movement

Due to the problems that may arise, forced stop events were added if it reached the extremes at different times than expected. These events were performed by software, using the options provided by the robot:



```
Archivo    Vacío    13:55:37    6305
Programa | Instalación | Mover | E/S | Registro
codigo_final_Gerson_v3
        set_digital_out(0, True )
      MoverJ
        Punto_de_paso_3
      Esperar DI[0]=HI
      set_digital_out(2, True )
      If activar_paso≟ True
        paso≔3
    Caso 19
      MoverJ
        Punto_de_paso_2
    Caso predeterminado
      If pausa≟ True
        stopj(2)
Evento digital_in[1]≟ True
  set_digital_out(2, True )
SubTarea_1
  Esperar: 1.0
  If var_1≟ True
    Bucle accion[0]≠1
      accion≔socket_read_ascii_float(1,"com_
      Esperar: 0.5
    activar_paso≔ False
    If accion[1]≟3
      parada≔ True
      set_digital_out(2, True )
      stopj(2)
    ElseIf accion[1]≟1
```

Comando | Gráficos | Estructura | Variables

Evento

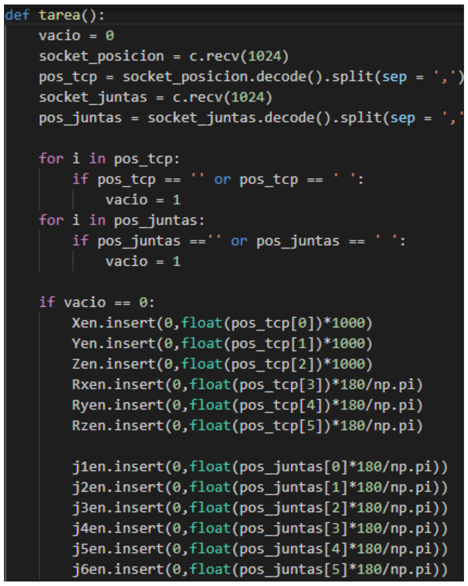
Un evento es similar a una interrupción, sin embargo, en un event
programa principal prosigue mientras que se ejecuta el código del
se ejecute el evento, los eventos nuevos no tendrán efecto.

En función del estado de la entrada del sensor o de la variable del
indicadas, se ejecutarán las siguientes líneas.

digital_in[1]≟ True

Fig 30. Logical condition for event triggering

Sending the current state of the robot after each movement, to be displayed in the text fields of the interface. This is important because in order to keep the robot monitored from the interface, it is necessary to constantly send the robot's position. This was done using the functions of obtaining the TCP position of the robot and the positions of the joints. This data is sent through the communications port to the code in Python, which is responsible for receiving, segmenting and organizing it:



```
def tarea():
    vacio = 0
    socket_posicion = c.recv(1024)
    pos_tcp = socket_posicion.decode().split(sep = ',')
    socket_juntas = c.recv(1024)
    pos_juntas = socket_juntas.decode().split(sep = ','

    for i in pos_tcp:
        if pos_tcp == '' or pos_tcp == ' ':
            vacio = 1
    for i in pos_juntas:
        if pos_juntas =='' or pos_juntas == ' ':
            vacio = 1

    if vacio == 0:
        Xen.insert(0,float(pos_tcp[0])*1000)
        Yen.insert(0,float(pos_tcp[1])*1000)
        Zen.insert(0,float(pos_tcp[2])*1000)
        Rxen.insert(0,float(pos_tcp[3])*180/np.pi)
        Ryen.insert(0,float(pos_tcp[4])*180/np.pi)
        Rzen.insert(0,float(pos_tcp[5])*180/np.pi)

        j1en.insert(0,float(pos_juntas[0]*180/np.pi))
        j2en.insert(0,float(pos_juntas[1]*180/np.pi))
        j3en.insert(0,float(pos_juntas[2]*180/np.pi))
        j4en.insert(0,float(pos_juntas[3]*180/np.pi))
        j5en.insert(0,float(pos_juntas[4]*180/np.pi))
        j6en.insert(0,float(pos_juntas[5]*180/np.pi))
```

Fig 31. Receiving data to be displayed in the interface

Figure 34 presents a flowchart that simplifies understanding of the process performed in the main program. However, it should be clarified that some of the processes are carried out simultaneously with the main program, such as the reception of the data of the position of the cobot, which is carried out periodically to be shown in the respective changes of the interface. Another activity that is carried out in parallel is the verification of the pause, continuation and emergency stop buttons, this thanks to the subtask that was discussed earlier, in Figure 27.

These changes were necessary to perform the correct validation tests, in addition to using the connections to digital inputs and outputs for the control section of the base (see figures 32 and 33), as well as using the 24 VDC output for the power supply (power section) of this.

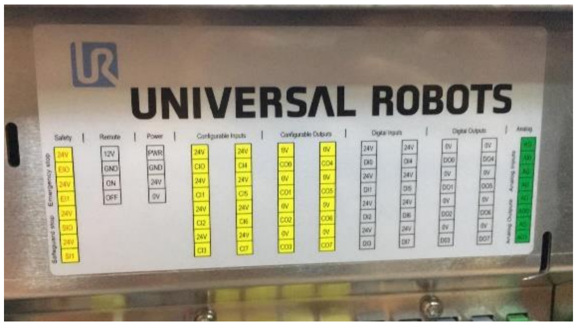


Fig 32. Diagram of the accessible terminals of the cobot controller

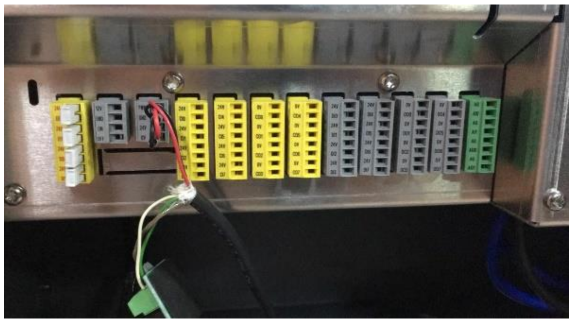


Fig 33. Accessible robot terminals

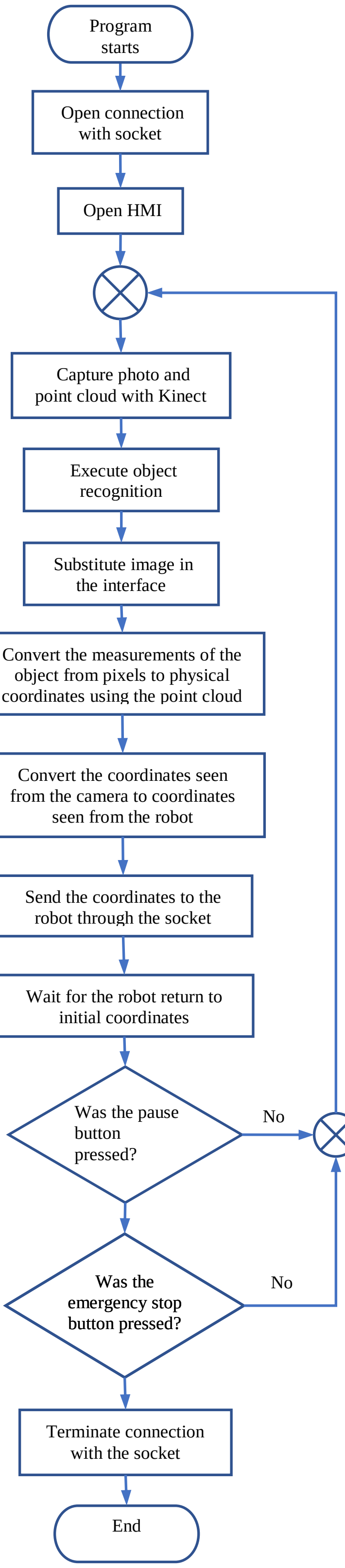

Program starts
Open connection with socket
Open HMI
Capture photo and point cloud with Kinect
Execute object recognition
Substitute image in the interface
Convert the measurements of the object from pixels to physical coordinates using the point cloud
Convert the coordinates seen from the camera to coordinates seen from the robot
Send the coordinates to the robot through the socket
Wait for the robot return to initial coordinates
Was the pause button pressed?
No
Was the emergency stop button pressed?
No
Terminate connection with the socket
End

Fig 34. Main program flowchart

8. CONCLUSIONS

• With the development of this project it was possible to demonstrate the easy handling and incorporation of a cobot in a workspace to facilitate processes and increase production times.
• To better check the operation of both the linear guide and the artificial vision system, it is better to do it in the real workspace, since when using a model there are many factors that affect and can vary the results and efficiency of the project.
• The best way to monitor the cobot from an external interface is to use a computer that has enough capacity to execute codes in parallel, taking photos and image recognition, sending and receiving data from the robot, and executing the internal code of the interface, since, in this case, there were many limitations when having to perform these processes sequentially.
• Although the minimum or maximum number of photos to develop a pre-trained model are not established, the time involved in taking photos as such must be taken into account, in addition to the time consumed by performing manual labeling. This leads to considering the balance between time and results of the models obtained.
• By using the controller provided by the robot, advantages were obtained such as: decreasing the number of connections needed, direct and reverse kinematic calculations, kinematic control of the robot, etc. Keeping all these elements separate from the program executed on the computer, allowed to distribute the computing capacity in the most appropriate way possible, keeping to a minimum the number of processes executed on the computer.

ACKNOWLEDGMENTS

This work is based on the capstone design project to qualify for the title of Mechatronics Engineer developed by Ortega and Castro (Ortega & Castro, 2020).

Bibliography

Bay, H., Tuytelaars, T., & Van Gool, L. (s.f.). SURF: Speeded Up Robust Features. En L. A., B. H., & P. A. (Edits.), *Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science* (Vol. 3951). Berlin: Springer. doi:https://doi.org/10.1007/11744023_32
Cuevas Jimenez, E. V., Zaldivar Navarro, D., Perez Cisneros, M. A., & Garcia Tome, A. (2010). *Procesamiento digital de imágenes con*

MATLAB y Simulink. RA-MA S.A. Editorial y Publicaciones.

Gonzalez Fernandez, V. R. (s.f.). *Estuctura de un robot industrial.* Recuperado el 21 de Junio de 2019, de http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/index.htm

Lozano Mantilla, G. A., & Orduz Rodriguez, J. J. (2015). *Diseño de un sistema de visión artificial para la revisión del nivel de llenado de bebidas embotelladas.* Universidad Autónoma del Caribe.

Ministerio de Salud y de la Protección Social: Resumen estadisticas Sistema General. (s.f.). Recuperado el 21 de Junio de 2019, de https://www.minsalud.gov.co/proteccionsocial/RiesgosLaborales/Paginas/indicadores.aspx

Norton, R. (2019). *Machine Design: An Integrated Approach* (6 ed.). Pearson.

Ortega, G., & Castro, J. (2020). *Desarrollo de un Prototipo a Escala para la Automatización de la Estación de Tratamientos Térmicos de las Campanas Exteriores de las Juntas Homocinéticas de la Empresa Transejes Utilizando Robótica Colaborativa y Visión Artificial.* Universidad Autónoma de Bucaramanga.

Pelegrí, J. (s.f.). *Beneficios de la automatización con robots colaborativos.* Recuperado el 21 de Junio de 2019, de https://blog.universal-robots.com/es/beneficios-robots-colaborativos

Pelegrí, J. (s.f.). *La robótica colaborativa en el sector de la locomoción.* Recuperado el 21 de Junio de 2019, de https://blog.universal-robots.com/es/robótica-colaborativa-automoción

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (págs. 779-788). IEEE.

Tabuenca Alcusón, D. (2017). *Implantación de robots colaborativos en linea de producción.* Universidad de Valladolid.

Universal Robots. (2016). *User Manual UR3.* Obtenido de https://www.universal-robots.com/download