Simulación, construcción y control de un robot Scara de 4 grados de libertad

H.V Cristian, M.H. José, G.S. Sergio, G.F. Miguel

Instituto Tecnológico de Toluca
Sistemas Robóticos, grupo 238400
M. Sc. Carlos Alberto Sánchez Delgado

Resumen—En este proyecto vamos a desarrollar el diseño de un robot Scara, el cual tiene 4 grados de libertad y se abordará la construcción física mediante el uso de impresión 3D, desarrollando los modelos en el software SolidWorks (SW) 2022v, además de la simulación y control mediante el uso de herramientas de Open Source, como lo son el Robots Operative System (ROS) y módulos de Python, todo software usado, menos SW, es de uso completamente gratuito; por lo que este es un excelente proyecto para estudiantes que buscan sus primeros acercamientos al mundo de la robótica, por lo cual es este trabajo es una herramienta que va a permitir al lector abrirse paso en el uso de estos programas.

Se obtuvo al final un robot Scara físico y que cumple en su programación con el Universal Robots Description Format (URDF), por lo que además puede ser simulado desde diferentes softwares.

***Abstract*— In this project we are going to develop the design of a Scara robot, which has 4 degrees of freedom and the physical construction will be addressed through the use of 3D printing, developing the models in the SolidWorks (SW) 2022v software, in addition to the simulation and control through the use of Open Source tools, such as the Robots Operating System (ROS) and Python modules, all software used, except SW, is completely free to use; so this is an excellent project for students seeking their first approaches to the world of robotics, which is why this work is a tool that will allow the reader to make their way in the use of these programs.**

In the end, a physical and functional Scara robot was obtained that complies in its programming with the Universal Robots Description Format (URDF), so it can also be simulated from different software.

I. Introducción

Este trabajo, hecho por alumnos del Instituto Tecnológico de Toluca (ITT) para la materia de Sistemas Robóticos de la carrera de Ingeniería Mecatrónica, trata el tema del diseño de un robot Scara, que cuenta con 4 grados de libertad y es hecho con la intención de la divulgación de las herramientas de Open Source disponibles de manera completamente gratuita ya que la robótica durante casi toda su vida ha sido un campo muy elitista en cuando a quienes pueden acceder a las herramientas necesarias para su desarrollo, siendo éstas en su mayoría programas con un alto costo de licencias para su uso. Por eso con este trabajo se podrá ver una parte del poder que tienen las herramientas gratuitas acerca de robots que están al alcance de todos y que aún no son tan difundidas como los softwares de propietario más comunes, además de que con el uso de la impresión 3D y el material Poliestireno Tereftalato Glicolizado (PETG). El trabajo está estructurado de manera que se pueda entender por qué se diseñó de una u otra manera cada componente, además de presentar la manera de manejar los distintos conceptos y programas usados a lo largo del mismo. Se busca un entendimiento común, tanto para lectores con experiencia en el campo de la robótica como para aquellos que este puede ser de sus primeros acercamientos, sino es que el primero a este campo del conocimiento, tan importante en la vida de nuestros tiempos. Como limitaciones principales de este trabajo debemos de enunciar que se necesita tener un cierto conocimiento previo sobre el lenguaje de programación de Python y conocimientos de matemáticas de Álgebra Lineal ya que se tratan mucho los temas de matrices de rotación y transformadas lineales. De cualquier manera, siempre se trata de explicar de la manera más clara y concisa las partes más importantes del trabajo, tanto del diseño como la programación y la simulación. Además de que todos los archivos del trabajo se encuentran contenidos en un repositorio de GitHub, el cual se encuentra el enlace en la sección de “Anexos”.

II. Marco Teórico

A. Manipuladores Robóticos: Morfología y tipos

Los robots llamados “manipuladores” suelen estar conformados por eslabones que, a su vez, están conectados mediante algo llamado “articulaciones” (un concepto general al que conocemos de las articulaciones de nuestro cuerpo), de este modo se les permite a los eslabones tener movimiento entre ellos mientras sean consecutivos uno de otro. [1]

Las articulaciones se pueden clasificar por el tipo de movimiento que tienen ya que existen diferentes tipos. Siendo estos tipos “desplazamiento” y “giro”, de hecho, puede haber articulaciones que tienen a la vez ambos tipos de movimientos.

A cada uno de estos movimientos (independientes) por cada articulación se le conoce como “grados de libertad” (GDL) y dependiendo del tipo de articulación del que estemos hablando, varía la cantidad de GDL pudiendo tener al menos un grado de libertad o más. Los tipos de articulación se muestran en la Fig. 1 para que se puedan dar una idea de lo que se está hablando en esta sección.

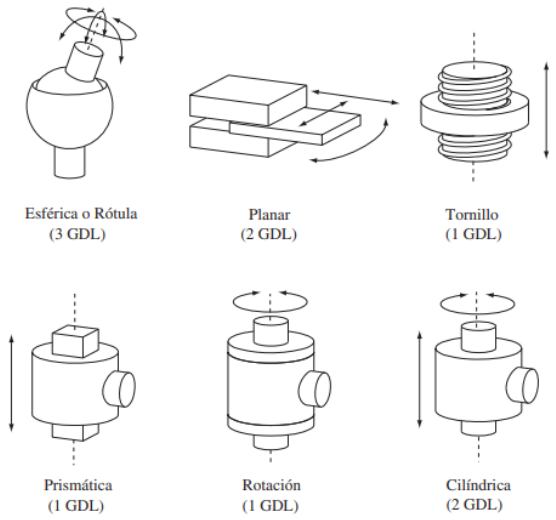

Fig. 1 Tipos de articulaciones existentes

Para efectos de la robótica con la que trabajaremos en este proyecto, solo vamos a considerar dos tipos de articulaciones. Las "Primáticas" (P) y las de "Rotación" (R). Por lo tanto, en los robots manipuladores casi siempre coincidirá el número de grados de libertar con el número de rotaciones pues todas las articulaciones son de un solo grado de libertad.

B. Tipos de robots manipuladores

Debido a que solo se utilizan articulaciones P y R, los tipos de robots manipuladores pueden denominarse con respecto a este tipo de articulaciones. Entonces, un robot que posea una articulación de rotación y dos prismáticas seguidas de ésta. A modo que el robot puede ser denominado un robot "RPP", teniendo 3 grados de libertad (claramente haciendo alusión a las articulaciones usadas en la construcción del robot). [2]

En la Fig. 2 se muestra las configuraciones más comunes de robots manipuladores, aquí podemos observar una clara diferencia entre las articulaciones P y R.

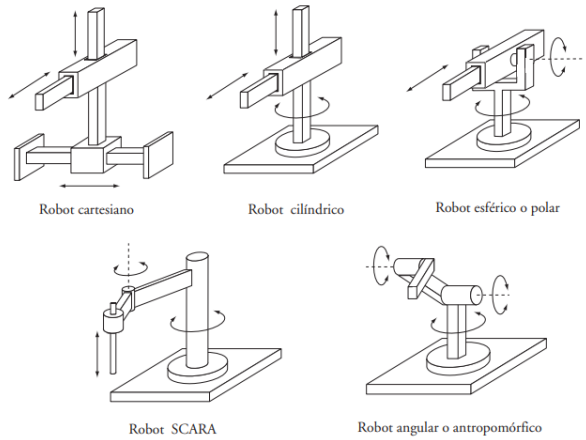

Fig. 2 Robots manipuladores más comunes

C. Robot SCARA

"Selective Compliance Assembly Robot Arm" es el nombre sin abreviar del robot SCARA, este suele tener 4 grados de libertad; fue desarrollado principalmente para trabajos de ensamble de piezas, por esto, se busca que el diseño del robot sea de coste limitado y que tuviera los 3 o 4 grados de libertad que comúnmente tienen.

Un robot SCARA suele ser de "accionamiento directo" o "Direct Drive" (DD), esto significa que los eslabones se conectan directamente por el eje actuador, sin la interacción previa de una caja reductora ni transmisiones de bandas como suele ocurrir en otras construcciones de robots. Con este tipo de accionamiento se logra tener una alta velocidad y una gran precisión. [3]

Gracias a esta falta de la caja reductora, los robots adquieren varias ventajas en ámbitos como el posicionamiento, haciéndolo así más veloz y preciso, así como una mayor controlabilidad en el sistema al evitar el juego entre los dientes de los engranes de la caja reductora; además de que al eliminar la reductora se pueden simplificar los modelos matemáticos de nuestro sistema.

Aunque el SCARA ha existido desde 1982, empezó a destacar hasta 1984 con la fabricación del primer robot de accionamiento directo de este tipo de robots. A partir de esta fecha, se empezó a popularizar su uso y hoy en día son usados en múltiples tareas que necesiten velocidad y precisión al mismo tiempo.

D. Unified Robot Description Format (URDF).

El URDF es un formato de lenguaje que sirve para describir robots, tantos sus componentes como sus propiedades. Esto se hace en el marco de la gramática Extensible Markup Language (XML), esto es un lenguaje de marcado, dentro del cual se definen un conjunto de reglas para la codificación de documentos.

Este lenguaje de marcado está formado por un conjunto de códigos que se pueden aplicar para analizar datos o incluso en la lectura de textos que han sido creados por personas o computadoras. En XML se tiene la posibilidad de crear un formato y un lenguaje personalizados sobre esta plataforma, como lo es el URDF.

El diseño que se puede ver en archivos XML va centrado hacia su simplicidad, generalidad y facilidad de uso. Entonces es visto su uso en varios servicios web, los XML no pueden confundirse con los HyperText Markup Language (HTML).

Entonces un URDF está escrito en un archivo XML y que por su gran versatilidad se puede usar para describir robots y éstos ser visualizados o manipulados desde diferentes simuladores, como Matlab, Pybullet o el Robot Operating System (ROS).

Una vez abordados los conceptos primordiales de un XML, vamos a seguir hablando del URDF en cuanto a su estructura. Recordemos que este formato lo vamos a usar para describir un robot (como su nombre lo dice), por lo tanto, podemos fácilmente deducir que en el URDF se deben de encontrar definidos elementos como los eslabones, articulaciones, sensores, etc. y también propiedades como lo pueden ser los momentos de inercia de los eslabones.

Vamos a comenzar retomando la estructura básica de un URDF (que incluya solo dos eslabones y una articulación), como se muestra en la Fig. 3

```
<robot>
    <link>

    </link>

    <link>

    </link>

    <joint>

    </joint>
</robot>
```

Fig. 3 Estructura básica de un URDF con dos eslabones y una articulación.

Vemos claramente como se utiliza una estructura con forma de árbol, propia de los archivos XML. En la Fig. 3 se muestra lo mínimo necesario para poder tener un robot con una articulación. Vemos como con la etiqueta "<robot>" definimos todo lo que forma parte de nuestro robot, de modo que todo lo que se encuentre entre esta etiqueta y el cierre ("</robot>") va a formar parte de nuestro robot. A partir de aquí vamos a considerar que el uso de apertura y cierre de etiquetas para definir la estructura del XML en forma de árbol ha quedado clara y no deberá ser explicada nuevamente para los demás casos.

En la Fig. 3 Estamos diciendo que nuestro robot contiene dos links (eslabones) y un joint (articulación)

Claro que los links tienen más etiquetas que pueden definir sus propiedades, como se puede encontrar en [4].

A continuación, vamos a mencionar las etiquetas más importantes que componen a los links y las joints.

Links

- Podemos definir un nombre a nuestro eslabón con una etiqueta con la forma "<link name=”nombreEslabon”>"
- "<visual>" define parámetros referentes a la visualización del eslabón.
    - "<origin/>" va a contener parámetros sobre el origen del eslabón.
    - "<geometry>" contiene los parámetros que hacen referencia a la ubicación del archivo .STL de nuestro eslabón y a la escala de este.
    - "<material>" va a contener lo referente al material del que se compone el eslabón.
- "<inertial>" en esta sección tendremos los parámetros de las propiedades inerciales de nuestro robot.
    - "<origin/>" nos define la posición del centro de masa respecto al marco de referencia del eslabón.
    - "<mass/>" con el parámetro "value" podemos definir una masa a nuestro eslabón.
    - "<inertia/>" en esta etiqueta vamos a pasarle como parámetros los valores de los momentos de inercia respecto a cada eje.

Joints

- Podemos definir un nombre y un tipo a nuestra articulación con una etiqueta con la forma "<joint name = ”nombreArticulacion” type = ”tipoArticulacion”>".
- "<parent/> define el eslabón padre de nuestra articulación.
- "<child/> define el eslabón hijo de nuestra articulación.
- "<axis/> nos permite definir el eje sobre el que esta articulación va a rotar, debe ser definido sobre el eje de referencia de esta articulación.
- "<limit/>" estipula los radianes que nuestra articulación rotacional va a tener como límites en su movimiento.
- "<origin/>" es la transformación del eslabón padre al hijo, la articulación está situada en el origen del eslabón hijo.

III. Desarrollo

A. *Diseño 3D*

Considerando el diseño primordial de un robot Scara, vamos a usar el software (de propietario) SW, con nuestra licencia estudiantil proporcionada por nuestra institución (ITT), para esto vamos a construir los 4 eslabones de los que se compone nuestro robot de tipo RRP.

Como este será un robot que construiremos de manera física, debemos de considerar en nuestro diseño medidas que podamos generar con el menor costo, pero que no por eso afecte a la funcionalidad de nuestro proyecto. Aunque ya fue mencionado con anterioridad, vamos a volver a mencionar que el medio de construcción del robot será la impresión 3D con material PETG, por lo que el área de impresión de nuestra impresora 3D debe de marcar una pauta para el diseño de todas las piezas que hagamos si queremos generarlas con ella, esta área de impresión es un cubo de lado 22 cm.

Para el primer eslabón, llamado lk_0, que sería nuestra base, tenemos la construcción que se muestra en la imagen Fig. 4.

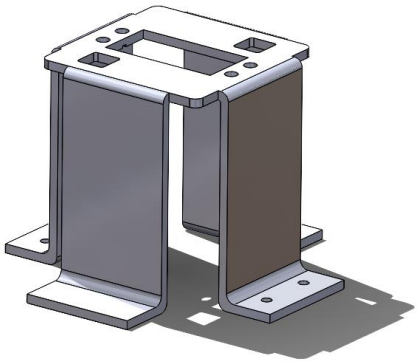

Fig. 4 Eslabón lk_0 del SCARA

Este primer elemento considera la presencia de un servomotor dentro de él, como se puede observar en la Fig. 5, la base está construida de modo que el servomotor puede ser fácilmente ensamblado en ella, además de permitir una conexión y ventilación impecables.

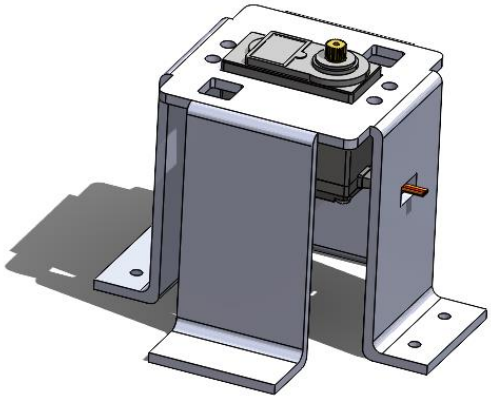

Fig. 5 Eslabón lk_0 del SCARA con el servomotor MG995 colocado

Pasando al siguiente eslabón, tenemos al llamado lk_1, este en conjunto con el eslabón lk_0 forman la primera articulación de nuestro robot, la cual es de tipo rotacional, vemos como en la Fig. 6 se muestra la parte inferior de este eslabón, la cual cuenta con una apertura para el acoplamiento mecánico entre el motor que se encontrará sujeto a lk_0 y va a hacer mover a nuestro lk_1.

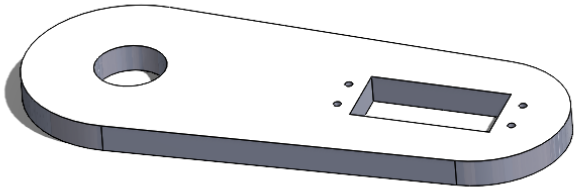

Fig. 6 Eslabón lk_1 del SCARA

Este eslabón describe una longitud de 100 mm entre el orificio visto en la parte izquierda de lk_1 y la parte derecha del mismo eslabón, la cual tiene el lugar donde el siguiente servomotor se va a colocar.

El elemento con el que se van a conectar los eslabones se puede ver en la Fig. 7.

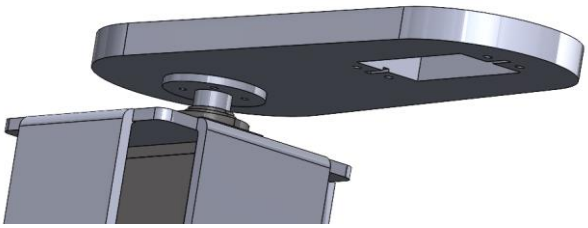

Fig. 7 Conexión de lk_0 y lk_1

El orificio que se encuentra en la parte derecha de lk_1 va a fungir como nueva conexión para la articulación formada entre este eslabón y el siguiente (lk_2) el cual se muestra su construcción en la Fig. 8.

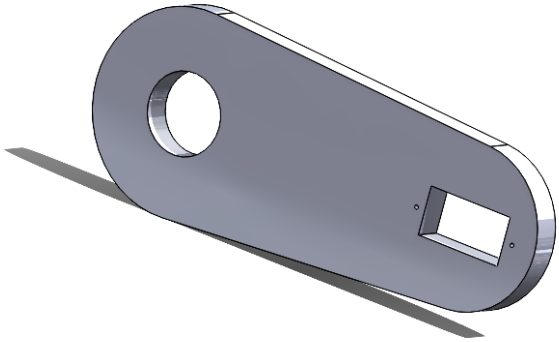

Fig. 8 Eslabón lk_2 del SCARA

La geometría de lk_2, nos va a permitir ahorrar mucho material en cuanto a la impresión de este, por lo que además de ahorrarnos tiempo de fabricación, estamos disminuyendo el momento de inercia provocado por la misma geometría del robot. Debemos de enforcarnos mucho en esto ya que al tener direct drive, nuestro robot tiene que superar los momentos de inercia provocados por la presencia de los servomotores.

Tanto en la Fig. 6 como en la Fig. 8 podemos observar que en ambos eslabones tenemos un agujero del lado izquierdo; esto es porque es ese lugar vamos a colocar rodamientos que faciliten el movimiento articular.

Por su parte, el agujero rectangular del lado derecho de lk_2 nos permitirá colocar ahí el servomotor que va a dar el último movimiento de rotación de nuestro Scara. El servomotor (MG90s, en azul) y el eslabón lk_2 se pueden ver ya ensamblados en la Fig. 9

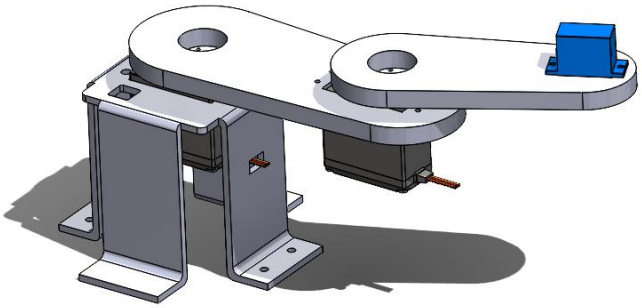

Fig. 9 Eslabón lk_2 y MG90s ensamblados

En este punto nos dimos cuenta de que para ayudar al eslabón lk_1 a soportar el momento creado por el segundo servomotor MG995 necesitábamos reforzar la geometría de este eslabón, y como vamos a utilizar los rodamientos, ocuparemos sus centros como otro punto de apoyo para tener una geometría más reforzada mediante el uso de un refuerzo, el cual podemos ver ensamblado en la Fig. 10.

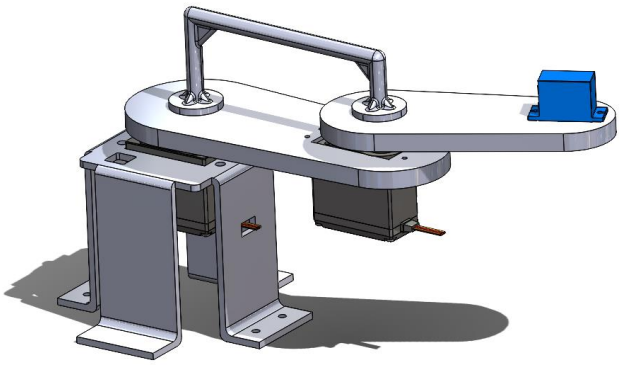

Fig. 10 Soporte colocado en el Scara

Como siguiente elemento, tenemos a lk_3, mostrado en la Fig. 11.

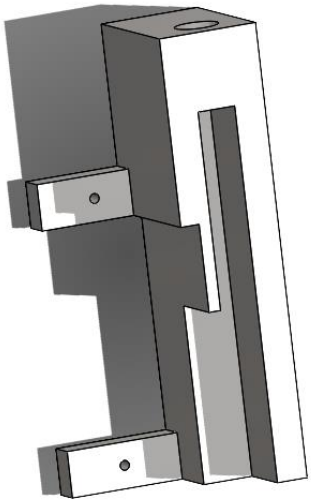

Fig. 11 Eslabón lk_3 del SCARA

Este eslabón va a ser el que presente el último movimiento rotacional de los eslabones del motor, y va anclado al servomotor MG90s, su ensamblaje se ve en la Fig. 12, donde también podemos observar como un segundo MG90s va anclado al eslabón lk_3

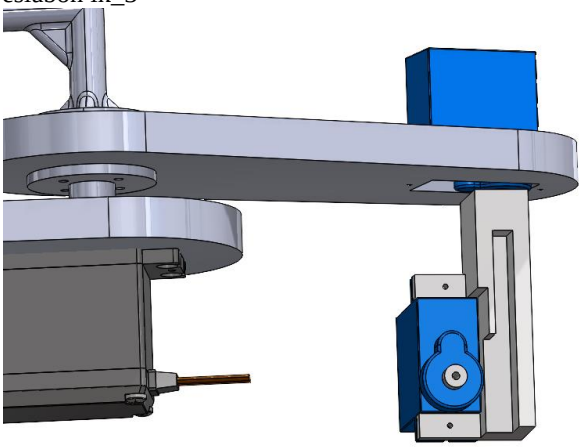

Fig. 12 Eslabón lk_3 y MG90s ensamblados

Como parte de este proyecto, la articulación final de nuestro Scara debe de ser una de tipo prismática, la cual conseguiremos al hacer un mecanismo de piñón-cremallera, el cual fungirá como nuestro último eslabón y a pesar de tener en su servomotor un movimiento de tipo rotacional, va a generar un desplazamiento prismática, este eslabón lk_4 puede ser visto en la Fig. 13

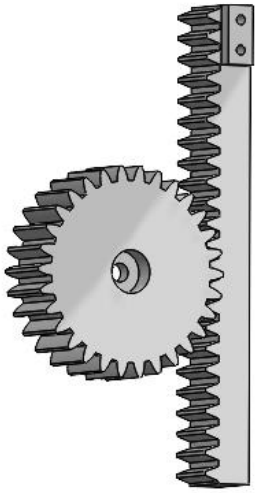

Fig. 13 Piñón-cremallera (eslabón lk_4)

La manera en la que se ensambla lk_4 al Scara se muestra en la Fig. 14

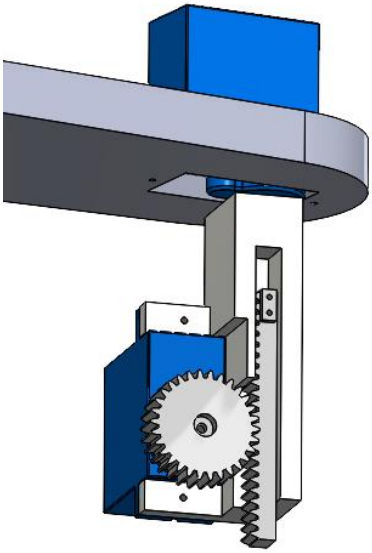

Fig. 14 Ensamble de lk_4

Quedando el ensamble de nuestro robot Scara como se muestra en las Fig. 15.

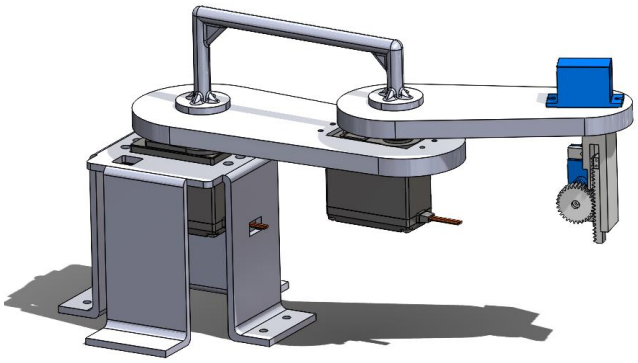

Fig. 15 Ensamble del SCARA

B. *URDF del Scara*

El URDF como se vio en la sección "II. Marco Teórico" referente a este apartado, es un archivo .XML que con su estructura vamos a definir los eslabones y las articulaciones de nuestro robot. De modo que con lo explicado en el marco teórico se puede entender lo que se presenta a continuación.

Debemos de dejar claro en este punto que este .urdf.xml debe de estar guardado en la misma carpeta que los .STL que exportamos de todos los eslabones de nuestro robot desde SW, de modo que facilita la escritura de las rutas de los .STL de los eslabones como se verá más adelante. Así se hizo en este proyecto y de hacerse de otra manera se tendrían que hacer los cambios correspondientes a las rutas de los eslabones en el .urdf.xml para que todo trabaje adecuadamente.

Vamos a ejemplificar como se hizo la configuración de los primeros dos eslabones y la articulación que los conecta y además la configuración de los últimos dos eslabones y su respectiva articulación, de modo que el lector pueda extrapolar fácilmente este ejemplo a lo necesario para hacer lo referente a las demás articulaciones y eslabones. Recordando que en la sección de "Anexos" se encuentra el enlace al repositorio de GitHub que tiene todos los sólidos, programas y archivos necesarios para este proyecto.

```
<?xml version="1.0"?>

<robot name="robot_scara">
    <!-- Linkages  -->
    <link name="lk0">
        <visual>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
            <geometry>
                <mesh filename="lk_0.STL" scale="1.0 1.0 1.0" />
            </geometry>
            <material name="black">
                <color rgba="0.4 0.4 0.4 1.0"/>
            </material>
        </visual>
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <mass value="1"/>
            <inertia ixx="1"  ixy="0"  ixz="0" iyy="1" iyz="0" izz="1" />
          </inertial>
    </link>

    <link name="lk1">
        <visual>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
            <geometry>
                <mesh filename="lk_1.STL" scale="1.0 1.0 1.0" />
            </geometry>
            <material name="yellow">
                <color rgba="1.0 1.0 0.0 1.0"/>
            </material>
        </visual>
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <mass value="1"/>
            <inertia ixx="1"  ixy="0"  ixz="0" iyy="1" iyz="0" izz="1" />
          </inertial>
    </link>
```

Fig. 16 Definición de los dos primeros eslabones en el URDF.

Lo referente a los dos primeros eslabones (visto en la Fig. 16) nos marca como tenemos que denominar dentro del .urdf.xml un elemento para que lo identifique como un eslabón, podemos ver cómo es repetitiva la estructura entre ambos eslabones, las partes más importantes y a destacar son que se le debe de pasar el nombre correcto del sólido en formato .STL que contiene la geometría del eslabón correspondiente. Nada nos limita a nombrar los eslabones de otra manera, pero es mejor mantener una estructura consistente para que sea más difícil confundirse y sobre todo que otras personas puedan leer con facilidad los archivos de nuestro proyecto y entenderlos.

```
    <!-- Joints -->
    <joint name="q1" type="revolute">
        <parent link="lk0"/>
        <child link="lk1"/>
        <axis xyz="0.0 0.0 1.0"/>
        <limit lower="-3.1416" upper="3.1416" effort="1.0" velocity="1.0"/>
        <origin xyz="0.010219613 0.0 0.0867" rpy="0.0 0.0 0.0" />
    </joint>
```

Fig. 17 Definición de la primera articulación en el URDF.

Para la configuración de la articulación q1 podemos ver la Fig. 17 que muestra como de igual manera como con los eslabones, debemos de nombrar a la articulación, y también decir de qué tipo es (recordemos que en este robot todas son articulaciones de rotación). A continuación, definimos al parent (padre) y al child (niño) de esta articulación. También definimos el eje sobre el que va a girar, que para todas las articulaciones en este robot va a ser el eje Z (axis) y en la etiqueta "limit" vamos a decirle los límites de giro que tiene nuestra articulación.

Para la etiqueta "origin" vamos a considerar la distancia (en el parámetro "xyz") que tiene el origen de nuestra articulación sobre el eje del eslabón anterior, en este caso es una distancia de 0.110 m, en el parámetro de "rpy" es las rotaciones que tiene nuestro eslabón actual medidas sobre dichos ejes del eslabón anterior.

En la Fig. 18 y Fig. 19 vemos lo necesario para configurar los dos últimos eslabones y su articulación correspondientemente; esto es un proceso repetitivo al anteriormente explicado.

```
    <link name="lk3">
        <visual>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
            <geometry>
                <mesh filename="lk_3.STL" scale="1.0 1.0 1.0" />
            </geometry>
            <material name="yellow">
                <color rgba="1.0 1.0 0.0 1.0"/>
            </material>
        </visual>
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <mass value="1"/>
            <inertia ixx="1"  ixy="0"  ixz="0" iyy="1" iyz="0" izz="1" />
          </inertial>
    </link>

    <link name="lk4">
        <visual>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
            <geometry>
                <mesh filename="lk_4.STL" scale="1.0 1.0 1.0" />
            </geometry>
            <material name="yellow">
                <color rgba="1.0 1.0 0.0 1.0"/>
            </material>
        </visual>
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <mass value="1"/>
            <inertia ixx="1"  ixy="0"  ixz="0" iyy="1" iyz="0" izz="1" />
          </inertial>
    </link>
```

Fig. 18 Definición de los dos últimos eslabones en el URDF.

```
    <joint name="q4" type="prismatic">
        <parent link="lk3"/>
        <child link="lk4"/>
        <axis xyz="0.0 0.0 1.0"/>
        <limit lower="-3.1416" upper="3.1416" effort="1.0" velocity="1.0"/>
        <origin xyz="0.0 0.0 -0.014200672" rpy="0.0 0.0 0.0" />
    </joint>

</robot>
```

Fig. 19 Definición de la última articulación en el URDF.

C. *Visualización del robot en Pybullet*

Para poder traer al simulador Pybullet lo que contiene nuestro URDF necesitamos crear un programa de Python, el cual debemos empezar con la importación de los módulos necesarios como se puede ver en la Fig. 20.

```
import numpy as np
import pybullet as pb
```

Fig. 20 Importación de los módulos necesarios.

Vemos como en este punto son necesarios los módulos "numpy" y "pybullet" en nuestro ambiente de Python. A continuación, se muestra el ejemplo de como crear un ambiente de Python e instalar ambos módulos usando el gestor de ambientes de Python llamado Anaconda, esto lo debemos de hacer desde el Anaconda Prompt (desde Windows) o desde la terminal (en Linux.).

```
# conda create -n *NombreAmbiente* Python=3
```

Con nuestro ambiente de Python en el que queramos trabajar activado, vamos a correr los siguientes comandos en nuestro Anaconda Prompt.

```
# pip install numpy
# pip install pybullet
```

Ahora que tenemos ambos módulos instalados en nuestro entorno de trabajo, escribimos el código visto en la Fig. 21, primero podemos ver como creamos un "client" que es una conexión de Pybullet con GUI.

```

#Making an instance of a physical client
client = pb.connect(pb.GUI)
```

Fig. 21 Creando una instancia de Pybullet.

En la sección de "# Simulation parameters", vista en la Fig. 22 se define la gravedad y un tiempo de simulación, para este punto vamos a definir el tiempo de simulación en cero debido a que solo nos interesa confirmar que se pueda visualizar el robot en este momento.

```

# Simulation parameters
pb.setGravity(0,0,-9.81)
pb.setRealTimeSimulation(0)

```

Fig. 22 Definición de los parámetros de simulación.

Para decirle dónde está nuestro .urdf.xml le pasamos la ruta al objeto "robot" que estamos haciendo una instancia de Pybullet cargando un URDF. La ruta que se escriba en el programa debe de ser la correcta y en el parámetro de "useFixedBase=1" estamos diciendo que la base se debe de quedar fija, o sea que no se caiga al vacío todo nuestro robot, se interpreta que la base es el primer eslabón. Todo esto se ve en la Fig. 23. En este punto ya el simulador se abrirá y mostrará nuestro robot.

```
#Charging the objects
robot = pb.loadURDF("URDF/reno.urdf.xml", useFixedBase=1)
```

Fig. 23 Cargando el URDF de nuestro robot.

Como el tiempo de simulación está definido como cero, debemos de hacer algo para mantener la ventana del simulador abierta, de modo que hacemos que el programa espere una entrada para que se desconecte con pb.disconnect(), esto se muestra en la Fig. 24.

```
#Ending the program
input('Press any key to stop...')
pb.disconnect()
```

Fig. 24 Fin del programa. Hacerlo esperar hasta que tecleemos algo para que se cierre el simulador.

Para evitar confusiones, el código se muestra completo en la Fig. 25.

```
import numpy as np
import pybullet as pb

#Making an instance of a physical client
client = pb.connect(pb.GUI)

# Simulation parameters
pb.setGravity(0,0,-9.81)
pb.setRealTimeSimulation(0)

#Charging the objects
robot = pb.loadURDF("URDF/reno.urdf.xml", useFixedBase=1)

#Ending the program
input('Press any key to stop...')
pb.disconnect()
```

Fig. 25 Código completo para correr la visualización del URDF en Pybullet.

Este código es guardado en la misma carpeta que la carpeta que contiene a los .STL y el .urdf.xml generados, ésta última carpeta es llamada "URDF" en nuestro caso.

Para correr este programa vamos a abrir de nuevo nuestro Anaconda Prompt (Windows) o nuestra terminal (Linux)

Utilizamos los siguientes comandos para correr la simulación:

```
# conda activate *NombreAmbiente*
```

Una vez activado el entorno de Python con el que estamos trabajando, vamos a utilizar el siguiente comando para ir a la carpeta que contiene el archivo de Python recién escrito y la carpeta que tiene el .urdf.xml y los .STL

```
# cd *RutaCarpeta*
```

Por ejemplo --- # cd Desktop\Scara --- (recordando que en Windows las rutas de carpetas van separadas con " \ " y en Linux con " / " .

Una vez dentro de esta ruta, vamos a correr el archivo de Python con el siguiente comando:

```
# python *NombreArchivoPython*
```

Por ejemplo --- # python visualize_scara.py ---

Al correr este comando podemos visualizar por fin nuestro robot en el simulador, el cual será mostrado en el apartado de "IV. Resultados".

D. Construcción del Scara en físico

Segunda fase:

Para poder construir nuestro Scara, que para este punto decidimos bautizar como "Reno", hemos hecho uso de la impresión 3D con filamento PETG además de los siguientes componentes:

- Arduino Mega
- Servomotor MG995 x2
- Servomotor MG90s x2
- Potenciómetro de 10KΩ x4
- Cables jumper
- 1 protoboard

La impresión de algunos de los componentes se puede observar en las Fig. 26 y Fig. 27

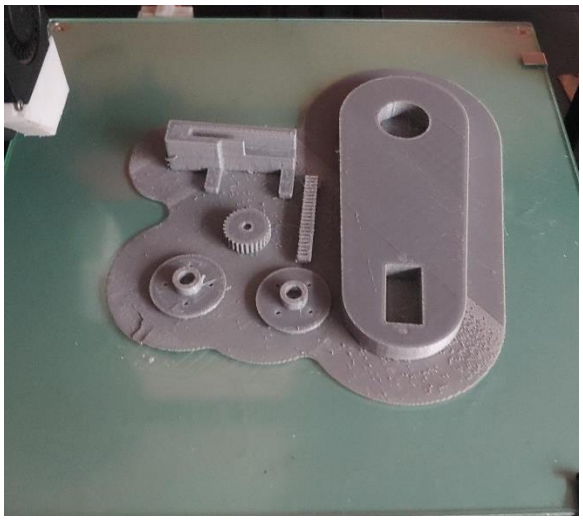

Fig. 26 Impresión en 3D (1)

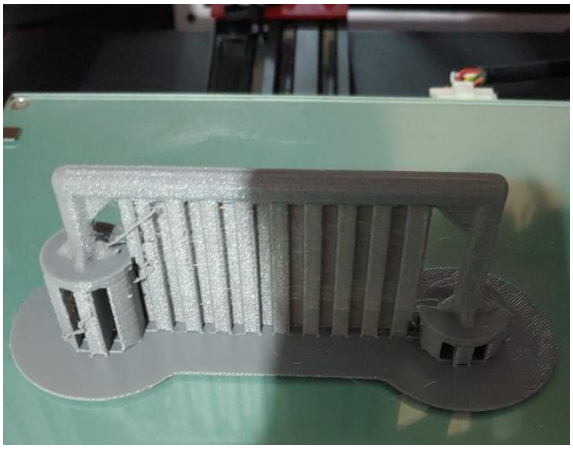

Fig. 27 Impresión en 3D (2)

Una vez contando con todos los elementos que necesitaran impresión de nuestro robot; procedimos a realizar todos los montajes y las conexiones necesarias, una muestra de un servomotor MG995 ensamblado en lk_0 puede ser visto en la Fig. 28

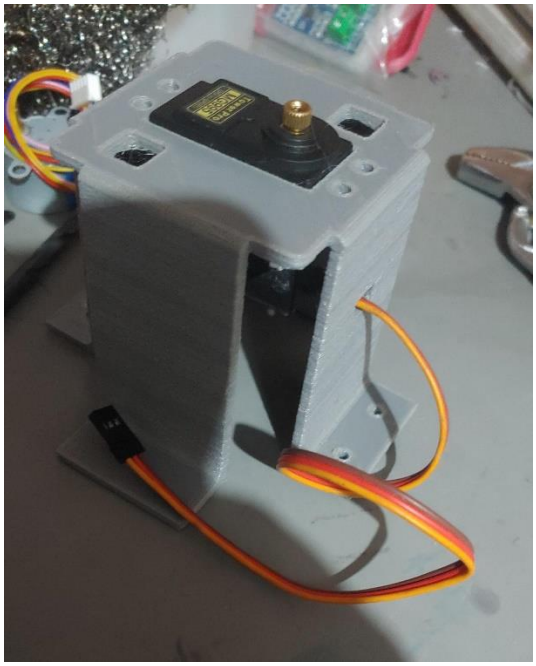

Fig. 28 Servomotor MG995 ensamblado en lk_0

La construcción física ya terminada de Reno es presentada en el apartado "IV. Resultados"

IV. Resultados

A. Construcción del modelo a computadora

Como parte fundamental, tenemos que hemos obtenido el diseño a computadora, también conocido como Computer-Aided Design (CAD) de nuestro robot Scara, el cual nos facilita la visualización del modelo sin la necesidad de planos físicos, además de su facilidad para la edición del mismo en caso de que algún cambio por corrección o mejora sea necesario. A continuación, en la Fig. 29 se puede ver el ensamblaje de nuestro Scara ya terminado, contando con los servomotores colocados en la posición en la que estarán físicamente.

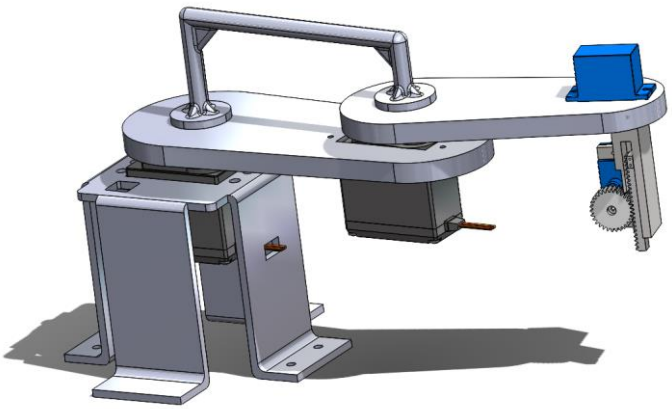

Fig. 29 Ensamble final del SCARA

B. *Visualización del URDF en Pybullet*

Cuando corramos los comandos vistos en la sección "Visualización del robot en Pybullet" del apartado III. Desarrollo, se nos abrirá el simulador con nuestro Scara visualizado en la pantalla, como prueba lo visto en Fig. 30 y Fig. 31.

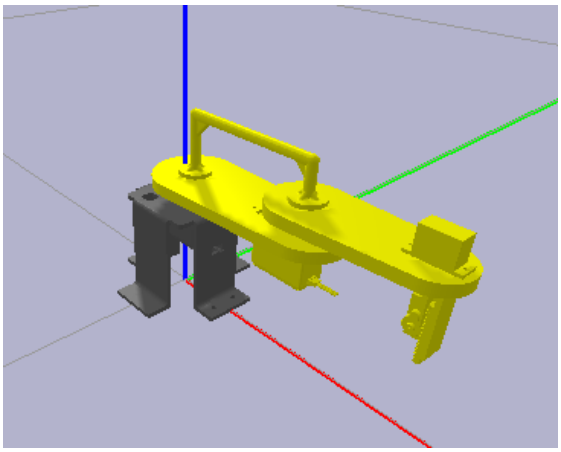

Fig. 30 Scara en Pybullet (1)

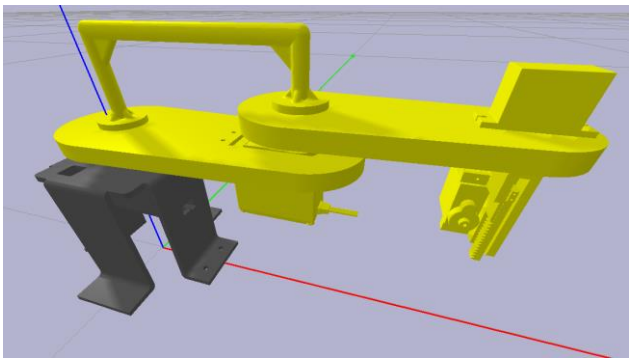

Fig. 31 Scara en Pybullet (2)

C. *Robot Scara en físico*

Segunda etapa:

Para esta segunda entrega, realizamos la construcción del robot Reno y como solamente se requiere de la comprobación de las geometrías y del funcionamiento correcto de nuestras articulaciones, vamos a operar los servomotores con un PWM causado por un Arduino Mega dependiendo de qué posición se le de con el potenciómetro; en la Fig. 32, Fig. 33 y fig. 34 podemos observar a nuestro Scara Reno ya contraído físicamente.

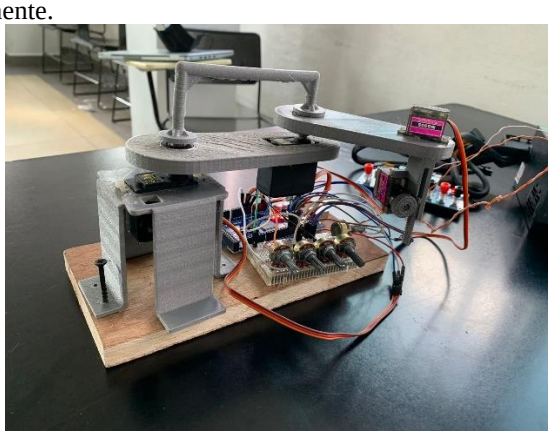

Fig. 32 Modelo físico del Scara Reno (1)

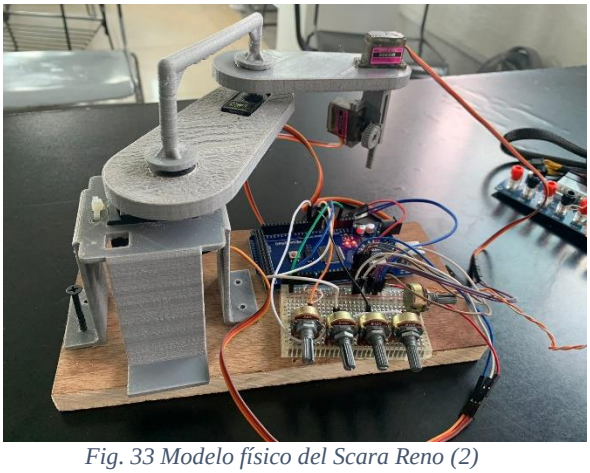

Fig. 33 Modelo físico del Scara Reno (2)

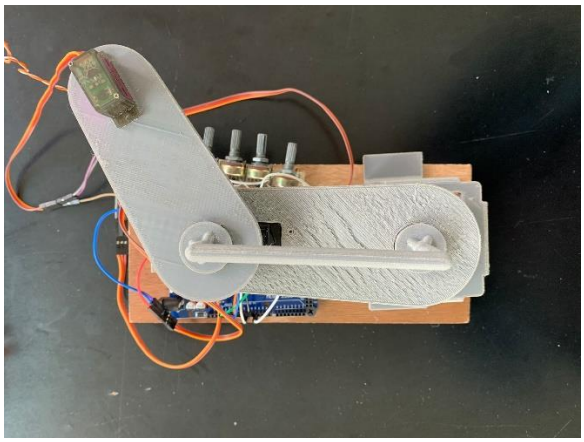

Fig. 34 Modelo físico del Scara Reno (3)

La comprobación del funcionamiento de los actuadores y de la correcta función de las geometrías es visto en el vídeo presentado para esta entrega.

V. Conclusiones

A. *Gutiérrez Figueroa Miguel*

Primera Etapa: Trabajando en la primera fase del proyecto pudimos observar a mayor detalle las estructuras de los robots y sobre todo analizarlas más a fondo, pues a la hora de realizar el diseño en 3D tuvimos que revisar cada aspecto para que saliera de forma correcta, esto debido a ciertas complicaciones que se nos fueron presentando a lo largo del trabajo pues no solo era hacer que tuviera la forma que deseamos, si no verificar que el diseño sirviera con todos los parámetros establecidos para el proyecto como el número de articulaciones y piezas. Así pues, después de trabajar cada aspecto del diseño, esperamos obtener mejores y más interesantes resultados en las siguientes fases a venir para poder poner en práctica conocimientos de electrónica y programación.

Segunda Etapa: En esta etapa pudimos notar mayores complicaciones a la hora de realizar esta parte del proyecto, principalmente el hecho de trasladar todo lo realizado en SolidWorks a un ensamblado real, primeramente tuvimos unas cuantas complicaciones pues ahora debimos tomar en cuenta toda la parte electrónica para que encajara perfectamente con nuestro diseño, teniendo incluso que modificar algunos

aspectos, del mismo modo, fue reconfortante observar cómo es que concordara el robot ensamblado de manera real con los softwares utilizados como lo son Pybullet y SW.

B. Gutiérrez Sánchez Sergio

Primera Etapa: Durante esta primera fase pude ver de forma más amílica las partes del robot, analizarlas y justamente realizar un diseño 3D del robot scara, justamente para adaptarlo a todos los componentes que tenemos planeados usar, y ver si cada uno de ellos se adaptaba al diseño que se está realizando.

Esta parte del proyecto es la más corta en cuestión de resultados, ya que hasta no tener pruebas de funcionamiento podremos ver si efectivamente nuestro diseño corresponde con lo que un robot scara debe hacer y con lo que debe tener como características. Sin embargo, la forma en que se planifica un proyecto como estos es la parte más importante, ya que desde aquí surgen todas las incógnitas a resolver para poder lograr un funcionamiento adecuado del robot.

Segunda Etapa: Durante el desarrollo de esta segunda etapa pudimos darnos cuenta de que el proceso si es un poco largo, pero no es complicado, claro, siempre y cuando el diseño inicial de nuestras piezas en el software cad pues esté realizado de forma correcta

Sin embargo, el proceso es tardado, ya que tuvimos que adaptar las piezas que ya teníamos a las medidas de los motores que íbamos a usar y a partir de eso, realizar los programas en Python además del URDF, en el cual encontramos un reto bastante grande al tener que adaptar nuestras piezas justamente a la simulación en pybullet

Pude darme cuenta de que es muy importante plantear tu robot desde el dibujo de las piezas, pasando por el proceso de exportado de las mismas para no encontrarte con algunos inconvenientes.

Sin embargo, es un proyecto al cual cada vez más le encontramos sentido, ya que con los conocimientos que estamos adquiriendo tanto para la simulación, aunado a lo que estamos llevando en la materia de robótica, todo cuadra y es más fácil poder visualizar más a detalle el funcionamiento del robot así como su construcción y simulación.

C. Hernández Villanueva Cristian Saúl

Primera Etapa: En esta primera etapa pude comprender la importancia de la planificación de los proyectos para construir cosas tan complejas como un robot, ya que desde las medidas hasta las orientaciones y los orígenes van a influir en facilitar o dificultar el posterior trabajo en nuestro proyecto, como al momento de escribir el URDF de este Scara o cuando queramos manipular la posición del robot con el uso de matrices de transformación homogénea. Además de que este fue uno de los primeros pasos para poder ver algo donde podamos aplicar conocimientos de muchos campos vistos en mi carrera.

Segunda Etapa: Esta etapa pareció más difícil de llevar a cabo debido a la complejidad que conlleva hacer un diseño en el que se tengan que tomar en cuenta cosas ya relacionadas con la vida real, tales como los presupuestos o hacer las cosas con geometrías que permitan un mejor ensamblado o una fácil impresión; esto junto con la parte electrónica y la programación de los primeros movimientos de nuestro robot ha sido lo más difícil de este proyecto hasta ahora. De igual manera puedo concluir que es importante conocer la conexión existente entre lo modelado en SW, lo impreso y ensamblado y lo simulado en Pybullet, como todo se relaciona haciendo que tengas que entender las demás partes para poder trabajar sobre una en específico.

D. Mena Huerta José Manuel

Primera Etapa: En esta primera fase del proyecto, nos topamos con las primeras dificultades del proyecto como lo son en el diseño y en la fabricación, por factores como los son; las dimensiones, las densidades, las utilidades, incluso las financieras. Con todo esto sumado a la parte electrónica y la programación que haremos en las siguientes fases el proyecto demuestra ser más interesante mientras se va enfocando más a los diversos conocimientos de nuestra carrera.

Segunda Etapa: En esta segunda entrega, en los aspectos de la estructuración del URDF, las correcciones en el modelado y la manufacturación de nuestro Scara fueron los aspectos más cruciales. En el URDF ordenar los marcos de referencia respecto al diseño CAD; en el diseño hacerlo loas simple y funcional posible; en la manufacturación fue en el ensamble y modificación de las piezas para un mejor acople para cumplir su función. Esos 3 aspectos fueron lo más fundamental para esta segunda etapa, en la que se empieza a percibir gran parte del proyecto final.

REFERENCIAS

[1] M. R, Cinemática y Dinámica de Robots Manipuladores, Ciudad de México: Alfaomega, 2016.

[2] A. Vivas, DISEÑO Y CONTROL DE ROBOTS INDUSTRIALES: TEORÍA Y PRÁCTICA, Clauca: Mc Grow Hill, 2019.

[3] M. G. W. &. T. S. Culebro, "Software libre vs software propietario: Ventajas y desventajas," Creative Commons., 30 Marzo 2006. [Online]. Available: https://www.mhe.es/cf/ciclos_informatica/8448180321/archivos/SOM_Legislacion_Software_libre_vs_software_propietario.pdf. [Accessed 15 Septiembre 2022].

[4] Open Robotics, "ros.org," ROS, 16 09 2022. [Online]. Available: http://wiki.ros.org/urdf/XML. [Accessed 15 10 2022].

[5] M. W. Spong, S. Hutchinson and M. Vidyasagar, Robot Dynamics and Control, New York: John Wiley & Sons, 2004.

Anexos

Repositorio de Github:
https://github.com/JellyfishDotcom/reno_scara_robot