









BRAZO ROBOTICO CILINDRICO

Integrantes:

-  Arias Ramos José Antonio Rey
-  Faraci Macías Salvador Alejandro
-  Ibarra Mercado Pedro Ignacio
-  Pinedo Serrano David Octavio
-  Solano Sandoval Jorge Alejandro
-  Zepeda Rosales Ana Yadira

Carrera: Mecatrónica

Grado/Grupo: 9ª

Profesor: Morán Garabito Carlos Enrique

Asignatura: Cinemática de Robots

Contenido

Objetivo general	3
Objetivos específicos	3
Justificación	3
Marco Teórico.....	4
Propuesta de materiales	6
Parametros Denavit-Hatemberg	6
Representacion grafica de los parametros D-H.....	8
Matriz D-H en MATLAB.....	9
Diseño del CAD	9
Análisis en ANSYS	10
Descripción del software	12
Conexiones del Hardware	15
Conclusiones	17
Anexos.....	21

Objetivo general

Construcción y diseño de un brazo mecánico con movimiento en los ejes X, Y, Z.

Objetivos específicos

- Desarrollar e implementar un control operativo mediante el sistema Ross, el cual dará ordenes al microcontrolador para convertirlas en acciones que finalmente realizará nuestro brazo.
- Diseñar e implementar la programación del sistema operativo ROS para la operación del microcontrolador.
- Construir la parte mecánica, la cual incluye el ensamblaje del brazo desde 0.
- Ensamblar y conectar la parte mecánica al ordenador.

Justificación

Reconocer las herramientas básicas que son necesarias para el armado y construcción de un robot; analizar los modelos matemáticos que son requeridos para utilizar los métodos geométricos necesarios para el desplazamiento y posicionamiento del robot dependiendo de los grados de libertad que este tenga.

Marco Teórico

El brazo cilíndrico es uno de los más sencillos de calcular. Los ejes **XYZ**, toman las variables **X** e **Y** para saber el resto de parámetros; el eje **Z** no interviene en el cálculo porque es en sí mismo un resultado. Se necesita calcular el ángulo de giro y el módulo (o también llamado radio). Esto nos recuerda al "sistema polar" visto el brazo desde arriba.

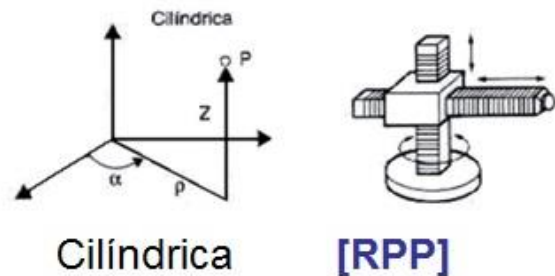


Imagen 1. Configuración del robot cilíndrico.

Robots de Configuración Cilíndrica (RPP)

La primera articulación es de tipo rotacional, produciendo por consiguiente rotación en torno a la base; en tanto que la segunda y tercera articulación es prismáticas. Es utilizado en operaciones de ensamblaje, manejo de máquinas-herramientas, soldaduras por puntos, y manejo, vaciado y moldeado de metales.

Un brazo robótico es un tipo de brazo mecánico, normalmente programable, con funciones parecidas a las de un brazo humano; este puede ser la suma total del mecanismo o puede ser parte de un robot más complejo.

Empleado para operaciones de ensamblaje, manipulación de máquinas herramientas, soldadura por punto y manipulación en máquinas de fundición a presión. Es un robot cuyos ejes forman un sistema de coordenadas cilíndricas.

Es una clase de robots que tienen movimiento rotacional en la base y dos ejes lineales perpendiculares, el segundo de ellos paralelo al de la base.

Usualmente este tipo de robots tiene una base rotativa, su primer segmento es capaz de deslizarse o extenderse hacia abajo o hacia arriba y lleva en su parte horizontal un segmento telescópico. Esta clase de robots son muy fácil de graficar y su envolvente de trabajo es muy intuitiva, pero son muy difíciles de implementar de manera efectiva ya que se requieren dos segmentos con movimientos lineales. En lo que es su distribución básica el código para su manipulación es relativamente simple.

Configuración TLO

Sus movimientos los realiza mediante coordenadas cilíndricas (α , ρ , z). Cuando las tareas a desarrollar o las máquinas servidas se encuentran alrededor del robot esta clase de robots son de gran ventaja.

Espacio de trabajo específico.

Las coordenadas cilíndricas son un sistema de coordenadas para definir la posición de un punto del espacio mediante un ángulo, una distancia con respecto a un eje y una altura en la dirección del eje.

Funcionamiento

Esta configuración de este tipo puede ser de interés en una célula flexible, con el robot situado en el centro de la célula sirviendo a diversas máquinas dispuestas radialmente a su alrededor.

Coordenadas cilíndricas: Son un sistema de coordenadas para definir la posición de un punto del espacio mediante un ángulo, una distancia con respecto a un eje y una altura en la dirección del eje.

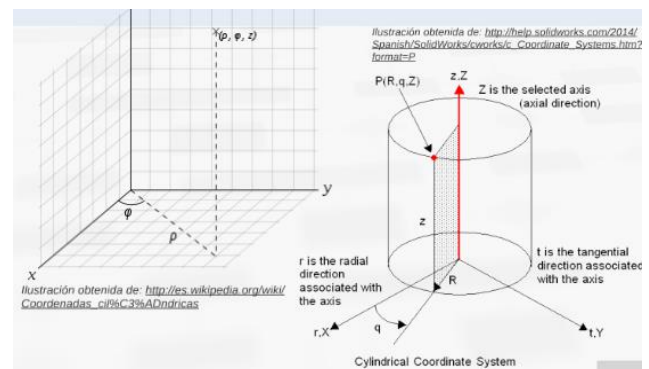


Figura 1.1. Representación de las coordenadas del robot cilíndrico.

Ventajas:

- Amplia área de trabajo por tamaño
- La programación es relativamente fácil
- Tiene la rigidez suficiente para manejar cargas pesadas a través espacios de trabajo de gran tamaño.

Propuesta de materiales

Cantidad	Material	Precio
4	Tuerca de Bronce para Husillo	\$110.20
4	Baleros: Rodamiento Lineal y Rodamiento para husillo	\$87
2	Esparrago de 8mm y 600mm de Largo	\$ 241.50
2	Varilla 8mm 50 cm Acero inoxidable lineal	\$ 84.50
2	Chumacera para sien mil one para tuerca 8mm	\$ 27.50
3	Motor a pasos nema 17	\$ 701
4	Chumacera KFL08 soporte	\$ 220
4	Chumacera con rodamiento 8mm	\$ 220
10	Madera 10cm x 20cm	\$ 150
3	Freescale	\$ 2400
3	Bases para Motor Nema 17	\$ 450
Total:		\$ 4,471.7

Tabla-1. Descripción de materiales utilizados, cantidades y precio.

Parametros Denavit-Hatemberg

Se trata de un procedimiento sistemático para describir la estructura cinemática de una cadena articulada constituida por articulaciones con.

Un solo grado de libertad. Para ello, a cada articulación se le asigna un Sistema de Referencia Local con origen en un punto Q_i y ejes ortonormales $\{X_i, Y_i, Z_i\}$, comenzando con un primer S.R fijo e inmóvil dado por los ejes $\{X_0, Y_0, Z_0\}$, anclado a un punto fijo Q_0 de la Base sobre la que está montada toda la estructura de la cadena. Este Sistema de Referencia no tiene por qué ser el Universal con origen en $(0, 0, 0)$ y la Base canónica.

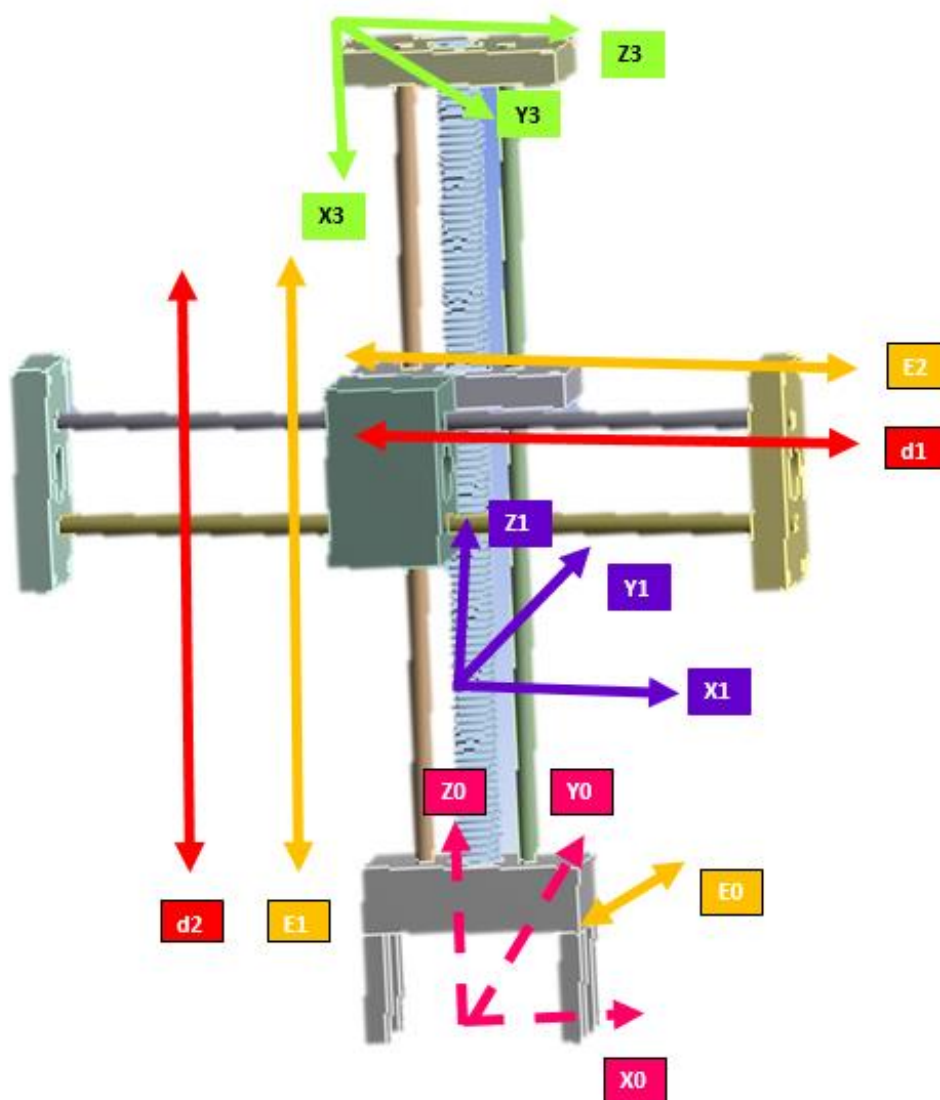
Los pasos del algoritmo genérico para la obtención de los parámetros DH se detallan a continuación:

1. **Numerar los eslabones:** se llamará “0” a la “tierra”, o base fija donde se ancla el robot. “1” el primer eslabón móvil, etc.
2. **Numerar las articulaciones:** La “1” será el primer grado de libertad, y “n” el último.
3. **Localizar el eje de cada articulación:** Para pares de revolución, será el eje de giro. Para prismáticos será el eje a lo largo del cuál se mueve el eslabón.
4. **Ejes Z:** Empezamos a colocar los sistemas XYZ. Situamos los Z_{i-1} en los ejes de las articulaciones i , con $i=1, \dots, n$. Es decir, Z_0 va sobre el eje de la 1ª articulación, Z_1 va sobre el eje del 2º grado de libertad, etc.
5. **Sistema de coordenadas 0:** Se sitúa el punto origen en cualquier punto a lo largo de Z_0 . La orientación de X_0 e Y_0 puede ser arbitraria, siempre que se respete evidentemente que XYZ sea un sistema dextrógiro.
6. **Resto de sistemas:** Para el resto de sistemas $i=1, \dots, N-1$, colocar el punto origen en la intersección de Z_i con la normal común a Z_i y Z_{i+1} . En caso de cortarse los dos ejes Z, colocarlo en ese punto de corte. En caso de ser paralelos, colocarlo en algún punto de la articulación $i+1$.
7. **Ejes X:** Cada X_i va en la dirección de la normal común a Z_{i-1} y Z_i , en la dirección de Z_{i-1} hacia Z_i .
8. **Ejes Y:** Una vez situados los ejes Z y X, los Y tienen su direcciones determinadas por la restricción de formar un XYZ dextrógiro.
9. **Sistema del extremo del robot:** El enésimo sistema XYZ se coloca en el extremo del robot (herramienta), con su eje Z paralelo a Z_{n-1} y X e Y en cualquier dirección válida.
10. **Ángulos teta:** Cada θ_i es el ángulo desde X_{i-1} hasta X_i girando alrededor de Z_i .
11. **Distancias d:** Cada d_i es la distancia desde el sistema XYZ $i-1$ hasta la intersección de las normales común de Z_{i-1} hacia Z_i , a lo largo de Z_{i-1} .
12. **Distancias a:** Cada a_i es la longitud de dicha normal común.
13. **Ángulos alfa:** Ángulo que hay que rotar Z_{i-1} para llegar a Z_i , rotando alrededor de X_i .
14. **Matrices individuales:** Cada eslabón define una matriz de transformación:

$${}^{i-1}\mathbf{A}_i = \left(\begin{array}{ccc|c} \cos \theta_i & -\cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cos \theta_i & -\sin \alpha_i \cos \theta_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array} \right)$$

Imagen 1.3. Matriz de transformación homogénea.

Representación grafica de los parametros D-H



Eslabon	$\alpha i - 1$	$\alpha i - 1$	d_i	θ_i
1	0	360°	0	θ_1
2	L	0	d1	0
3	40 L	0	d2	0

Tabla-2. Matriz Homogénea de los parámetros D-H

Matriz D-H en MATLAB

La obtención de la matriz homogénea del brazo cilíndrico se calculó mediante el software de Matlab, donde se realizaron las multiplicaciones correspondientes de la matriz de cada eslabón mediante la siguiente línea de código:

```
syms cteta1;
syms steta1;
syms dL;
syms d2;
t1= [cteta1, -steta1,0,0; 0, cteta1,0,0; 0,0,1,0; 0,0,0,1];
t2= [0,0,0,30;0,0,0,0;0,0,0, dL; 0,0,0,1];
t3= [0,0,0,40; 0, 0, 0,0; 0,0,0, d2; 0,0,0,1];
t4= t3*t2;
t5= t4*t1
```

La matriz homogénea es el resultado de la multiplicación entre las matrices 2_1 y 3_2 y su respectivo resultado 3_2 es multiplicado por la matriz 0_1 donde finalmente obtendremos la matriz homogénea 0_3

$$T^0_1 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ 0 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T^1_2 = \begin{bmatrix} 0 & 0 & 0 & 30L \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T^2_3 = \begin{bmatrix} 0 & 0 & 0 & 40L \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T^0_3 = \begin{bmatrix} 0 & 0 & 0 & 40 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Diseño del CAD

En base a estructuras geométricas primarias como líneas, puntos, arcos y polígonos diferentes, recreando una estructura en 3D de un modelo real basado en cortes, extrusiones que permiten el acoplamiento de los sistemas mecánicos, electrónicos e interfaces con las cuales el brazo robótico lograra realizar los movimientos correspondientes en cada eje X, Y, Z. (ver estructura física en el anexo 1).

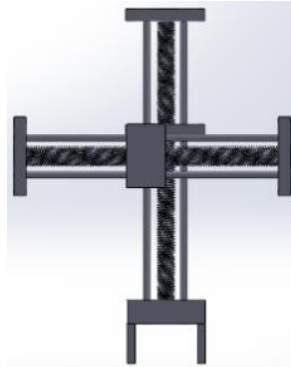


Figura 1.4 Diseño asistido por computadora (CAD).

Análisis en ANSYS

Es un ecosistemas de programas (CAE¹) para diseño, análisis y simulación de partes por elementos finitos ²(FEA), incluyendo las fases de preparación, mallado, ejecución y post proceso, donde el programa ejecuta análisis de piezas sometidas a diferentes fenómenos físicos. (Ver anexo 2).

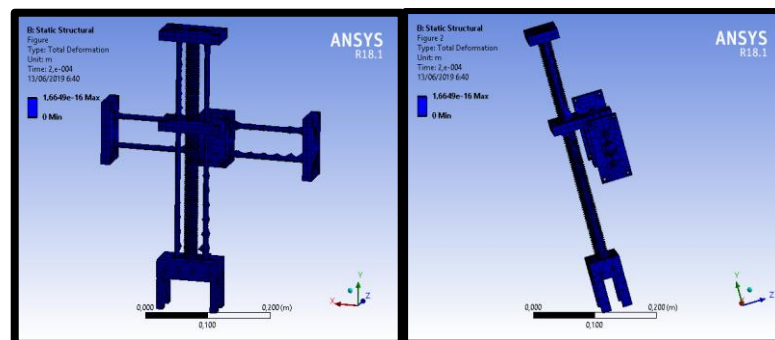


Figura 1.5 Solución de la deformación total de la estructura estática.

¹ **CAE:** Ingeniería asistida por ordenador (Computer-Aided- Engineering)

² **Elementos finitos:** Es un método numérico general para la aproximación de soluciones de ecuaciones diferenciales parciales muy complejas utilizado en diversos problemas de ingeniería y física.

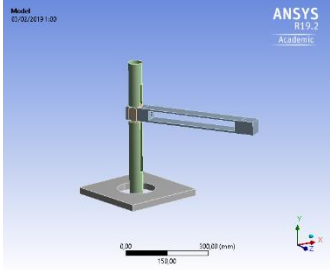
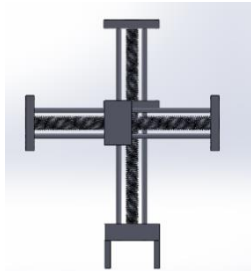
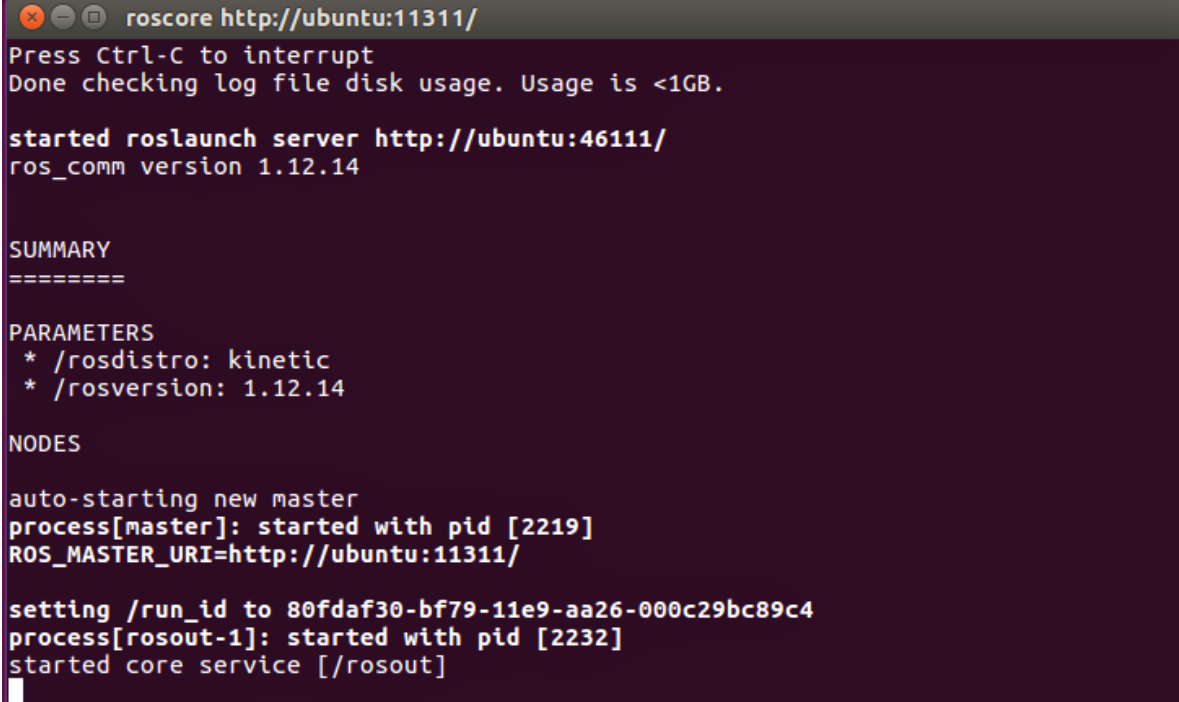
Diseño anterior		Diseño actual	¿Por qué se realizó el cambio?
			Se tuvieron dificultades para agregar el tercer grado de libertad, por lo que se modificó la estructura por completo con el fin de poder añadir el tercer grado de libertad.
Primer grado de libertad			
Base	La base contenía un balero, el cual haría rotar la estructura completa.	En el diseño actual la base se mantiene con el balero de 2 ½ in la cual hará rotar la estructura completa, sin embargo, se hará de un material más pesado para que pueda soportar el peso de la estructura.	Al cambiar el diseño de la estructura completa las nuevas piezas contienen pesos diferentes, por ello se tiene que colocar una base que sea capaz de soportar la estructura.
Segundo grado de libertad			
Esparrago de 35 cm	El esparrago funciona para la transmisión del movimiento entre la estructura que se mantendrá girando, la base y el 3er GDL.	Se realizaron modificaciones en la parte estructural externa del segundo grado de libertad. Al diseño se añadieron dos tubos acerados con sus respectivas bases.	No se agregó el 3er grado de libertad a la estructura externa, por lo cual se eliminó para poder ajustar la estructura del tercer grado de libertad.
Soporte del eje 3	En este soporte se buscaba agregar la estructura para el tercer grado de libertad.	Se modificó la estructura con un par de tubos acerados conectados a dos bases (superior e inferior) formarían una estructura capaz de soportar la estructura del 3er grado de libertad.	Esto servirá para que el tercer grado de libertad pueda desplazarse a través de los ejes X, Y, y Z respectivamente.
Tercer grado de libertad			
Esparrago de 40 cm	Se contaba solo con el material para realizar dicho grado, mas no logro implementarse.	Se pretende utilizar una estructura similar, con tubos acerados y sus respectivas bases conectados hacia el soporte del eje 3er con el de que pueda desplazarse hacia arriba, abajo, izquierda y derecha según corresponde.	Al agregar dicha estructura es posible agregar el tercer GDL con todo y los desplazamientos correspondientes.

Tabla 3. Tabla de ajustes realizados en la infraestructura del brazo cilíndrico.

Descripción del software

Uno de los puntos claves de la programación es el uso de los nodos por medio del sistema operativo Ros³, este sistema operativo trabaja en la programación por medio del envío de datos mediante la terminal⁴ del sistema.

Para comenzar a trabajar en el entorno de ros dentro de la terminal utilizamos el comando: **roscore**⁵.

A screenshot of a terminal window with a dark background. The title bar shows a window icon, a close button, and the text 'roscore http://ubuntu:11311/'. The terminal output is as follows:

```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:46111/
ros_comm version 1.12.14

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES

auto-starting new master
process[master]: started with pid [2219]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 80fdaf30-bf79-11e9-aa26-000c29bc89c4
process[rosout-1]: started with pid [2232]
started core service [/rosout]
```

Figura 1.6. Inicialización del sistema Roscore

³ **Robot Operating System:** es un sistema operativo que provee librerías y herramientas para ayudar a los desarrolladores de software a crear aplicaciones para robots.

⁴ **Terminal:** Todo dispositivo electrónico que forma parte del Hardware de un ordenador y que tiene la funcionalidad básica de ingresar los datos que se encuentran dentro de una computadora.

⁵**Roscore:** Es el nodo maestro el cual permite la interacción de otros nodos.

Para establecer la conexión entre el sistema operativo ros y el microcontrolador se utiliza el comando: `dmesg | grep tty`.

Este comando nos permite visualizar los puertos seriales ⁶en los que se puede establecer la comunicación serial entre el nodo⁷ y el microcontrolador.

```
salvadorfaraci@ubuntu: ~  
[1]+  Stopped                  rosrn rosserial_python serial_node.py /dev/ttyACM0  
salvadorfaraci@ubuntu:~$ dmesg | grep tty  
[ 0.004000] console [tty0] enabled  
[ 1.384962] 00:05: ttyS2 at I/O 0x3e8 (irq = 5, base_baud = 115200) is a 16550A  
[ 1.410712] 00:06: ttyS0 at I/O 0x3f8 (irq = 4, base_baud = 115200) is a 16550A  
[ 1.437476] 00:07: ttyS1 at I/O 0x2f8 (irq = 3, base_baud = 115200) is a 16550A  
[ 4.922067] cdc_acm 2-2.2:1.0: ttyACM0: USB ACM device  
[ 69.133740] cdc_acm 2-2.3:1.0: ttyACM0: USB ACM device  
[ 1171.224067] cdc_acm 2-2.2:1.0: ttyACM1: USB ACM device
```

Figura 1.7 visualización de puerto serial.

Una vez localizado el puerto al cual se encuentra conectado nuestro nodo utilizamos el comando: `roslaunch rosserial_python serial_node.py /dev/ttyACM1`

Dicho comando permite la conexión entre la tarjeta y el sistema.

```
salvadorfaraci@ubuntu:~$ roslaunch rosserial_python serial_node.py /dev/ttyACM1  
[INFO] [1565901915.359315]: ROS Serial Python Node  
[INFO] [1565901915.364994]: Connecting to /dev/ttyACM1 at 57600 baud  
[INFO] [1565901917.511424]: Note: publish buffer size is 512 bytes  
[INFO] [1565901917.512314]: Setup publisher on pot [std_msgs/Int16]  
[INFO] [1565901917.521233]: Note: subscribe buffer size is 512 bytes  
[INFO] [1565901917.521800]: Setup subscriber on servo [std_msgs/Int16]  
[INFO] [1565901917.530886]: Setup subscriber on servo2 [std_msgs/Int16]  
[INFO] [1565901917.547055]: Setup subscriber on servo1 [std_msgs/Int16]  
[INFO] [1565901917.584969]: Setup subscriber on varios [geometry_msgs/Twist]  
[ ]
```

Figura 1.8 conexión establecida por nodos.

⁶ **Puerto serial:** es una interfaz de comunicaciones de datos digitales más específicamente entre un ordenador y un periférico.

⁷ **Nodo:** Es un programa ejecutable en ROS, dedicado a procesar datos, los nodos se comunican entre sin mediante el uso de mensajes y topic con el fin de llevar a un objetivo en común.

Una vez realizado la comunicación con los nodos comenzaremos a trabajar con el microcontrolador freescale, primeramente agregando las librerías que se requiere utilizar para la comunicación usando el comando: `#include <libreria.h>`.

```
1 #include "mbed.h"
2 #include <ros.h>
3 #include <std_msgs/Int16.h>
4 #include <std_msgs/Float64.h>
5 #include <geometry_msgs/Twist.h>
6
```

Figura 1.9 librerías utilizadas en freescale

Para seleccionar los pines del microcontrolador al cual irán conectados nuestros motores utilizamos: `DigitalOut step (D2); DigitalOut dir (D5);`

Ahora se realiza el envío de datos mediante la ejecución del comando:

`Void servo_cb (const std_msgs::Int16& cmd_msg)` este comando se activa al ejecutar un orden en roscore.

Las condiciones para poder realizar los desplazamientos en X, Y, Z se ejecutan con los comandos:

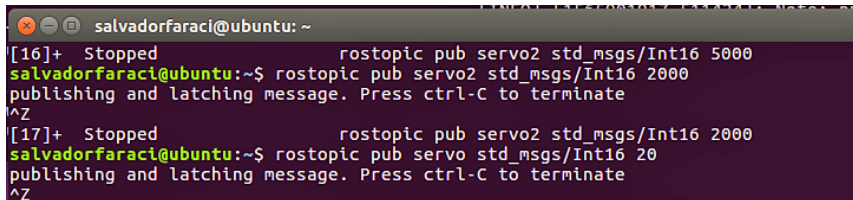
```
Void varios (const geometry_msgs: Twist& msg)
{
    servo_1 = msg.linear.x;
}
```

Las condiciones de giro de los motores se establecen mediante el comando:

```
If (servo_1>paso) {
    step=0;
    dir=0;
    en=0;
    wait(stepDelay);
    step=1;
    wait(stepDelay);
    paso=paso+1;
}
```

Para el posicionamiento de los motores trabajaremos en la terminal de ros (roscore tiene que estar ejecución) ahí utilizaremos el comando:

Rostopic pub servo 2 std_msgs/Int 16 donde agregaremos el número de pasos que el motor realizara.



```
salvadorfaraci@ubuntu: ~  
[16]+ Stopped          rostopic pub servo2 std_msgs/Int16 5000  
salvadorfaraci@ubuntu:~$ rostopic pub servo2 std_msgs/Int16 2000  
publishing and latching message. Press ctrl-C to terminate  
^Z  
[17]+ Stopped          rostopic pub servo2 std_msgs/Int16 2000  
salvadorfaraci@ubuntu:~$ rostopic pub servo std_msgs/Int16 20  
publishing and latching message. Press ctrl-C to terminate  
^Z
```

Figura 2.0 ejecución del comando para indicar el número de pasos del motor.

Conexiones del Hardware

Para la realización de las conexiones físicas (brazo cilíndrico de 3 ejes “X, Y, Z”) se utilizaron 2 tarjetas electrónicas una es la “Freescale” en la que se les ingresado el código y su ejecución y la otra es la tarjeta “CNC SHIELD” para el movimiento de los tres ejes con ayuda de los motores “NEMA-17” ya que la tarjeta SHIELD ya cuenta con la estructura de adaptación para la tarjeta Freescale.

Al realizar la adaptación se implementó una etapa de potencia para proteger a ambas tarjetas por lo cual se optó por usar un multímetro ya que este cuenta con un fusible para cualquier sobrecarga, realizando un puente para el chequeo de amperaje entre la fuente y la tarjeta SHIELD.

Para alimentar a la tarjeta se usó un voltaje de entre 12v – 36v y los drivers son de manera predefinida y estos cuentan con una conexión de 4 partes, para conectar los motores, lo que quiere decir 2 por bobina ya que estos son unipolares sus 4 pines.

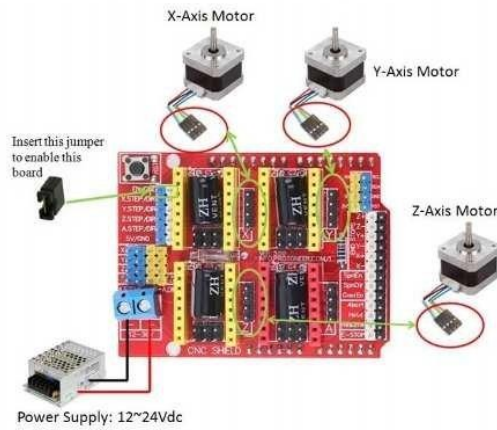


Figura 2.1 conexión de los motores NEMA 17" con la CNC shield.

El microcontrolador con el cual se realizara la comunicación es la freescale la cual cuenta con el Compilador "MBED", utilizado para uC's con arquitectura ARM.⁸ La freescale cuenta con:

- 32 Bits ARM Cortex M0+ Core.
- 128KB de flash.
- 16KB de SRAM.
- Frecuencia de operacion hasta 48 Mhz.
- USB OTG de alta velocidad.
- OpenSDA USB program and debug interfaz.
- LED RGB PWM para usuario.
- Acelerómetro 3 ejes digital integrado.
- Voltaje de operacion 1.71V - 3.6V.
- Voltaje de entrada 4.3V - 9V.
- Botón de reset.

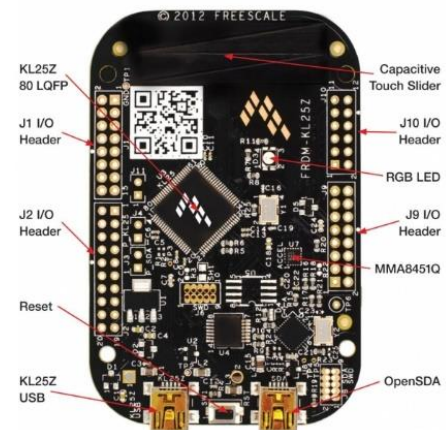


Figura 2.2 microcontrolador Freescale

⁸ Un procesador ARM es un procesador que está basado en la arquitectura RISC (Reduced Instruction Set Computer).

Conclusiones

Pedro Ibarra:

Se presentaron diversas problemáticas en el desarrollo del brazo robótico, tanto en la forma física como la programable, con los cuidados de estética que debe tener el brazo robótico pedidos por el profesor, la complicación para llevar a cabo este proyecto son los diseños, ya que hay un sinfín de formas de realizar el mismo brazo robótico y con diversos tipos de materiales, dentro de este hay que tener en cuenta el peso y la estabilidad del brazo robótico.

Después de elegir los materiales y un diseño el cual tenga pruebas validadas y evaluadas por un software como ANSYS, sigue la construcción de este de forma física, cuyo caso se convierte en una tarea difícil de realizar ya que los diseños siempre serán pruebas hasta lograr el más estable posible a tus necesidades, que es la parte en la que más complicada se volvió para nosotros de poder cumplir, ya que estuvimos teniendo muchos errores de diseño, hasta haber logrado el más estable.

La programación y el diseño de este también fueron una parte muy complicada y de las más importantes y que por ello se han realizado muchas pruebas para lograr nuestro objetivo de hacer un brazo robótico. Con esto he adquirido un conocimiento increíble de poder realizar y programar un brazo robótico que tienden a ser muy usados en las industrias, espero disfrutar la aplicación de estos conocimientos y aprovechar una buena oportunidad de seguir desarrollando proyectos como estos.

David Pinedo

A lo largo de la revolución industrial se han tenido diversas modificaciones en toda la industria para la automatización de procesos, de los cuales la ayuda de brazos robotizados o brazos robóticos como comúnmente se les conoce, estos han tenido grandes avances, innovaciones y además una división de grupos para desarrollar tareas específicas para poder mejorar la producción, dentro del proyecto se tuvieron diversas dificultades que fueron desde el diseño hasta el código en sí ya que en todo se tuvieron que hacer unas pequeñas modificaciones de las cuales fuimos aprendiendo que es lo que mejor se tenía

que acomodar o incluso cambiar al final todo resultado de una forma satisfactoria, con el robot en una condición aceptable.

Alejandro Faraci

Este gran proyecto ha sido uno de los mayores retos en cuestión de proyectos junto con el JAPDRA, por lo que el cálculo, decisión de materiales, la armadura, piezas faltantes, contratiempos etc. ha sido algo muy difícil pero no imposible, aún queda detalles por perfeccionar.

En la parte de programación y potencia se ha tenido ciertas dificultades porque se han quemado tarjetas, entre ellas una freescale y PSoC pero de los errores se aprende, actualmente ya tenemos motores funcionando con su cnc shield sin comprometer la tarjeta.

Rey Arias

Para la realización de este proyecto, involucra todas mis materias cursadas a lo largo de mi carrera, reforzando las áreas de diseño mecánico, programación, y mecánica. Ya que nos tocó llevar a cabo la realización de un “brazo cilíndrico” por lo cual consta de 3 eslabones, 3 grados de libertad que son los ejes “x”, “y”, “z”.

El cual este consiste en tener 3 tipos de movimiento constando desde el eje **z** con una rotación de 360°, después con el eje **y** deslizándose de manera vertical y finalmente el eje **x**, con el que se desplaza de manera horizontal.

Principalmente nos centramos en el material para el brazo, el cual consta de madera con ello se realizaron algunas bases para poder poner los baleros, rodamientos, coplees, motores, esparrago, y tubos que sirven para guías del mismo, con el armado se tuvo un poco de problema ya que accidentalmente no quedaban centrados los componentes en la madera, por lo que nos dimos a la tarea de trazar una cuadrícula en ella para tener una mejor precisión de las cosas.

Después se realizó la parte de la programación para poder mover los tres motores con su respectivo eje, en ellos si hubo un poco de complicación ya que no presentaba mayor torque para dicho movimiento, pero finalmente resultó.

Creo que este ha sido uno de los más grandes retos que se ha tenido durante estos años ya que el hacer el armado y la programación de un robot tiene su grado de dificultad, ya que aparte de hacer lo mencionado, también se necesitan hacer, cálculos matemáticos de cada eje, peso, fuerza o torque, hacer el diseño en algún CAD, hacer pruebas, contar con un buen presupuesto, pero al final del día es una gran experiencia para cada uno de nosotros ya que con ello.

Jorge Solano

La complicación más evidente que nos causó muchos retardos fue armar el robot, no teníamos como realizar las medidas exactas de los dispositivos móviles, el cual cuadrar todos los eslabones nos llevó más tiempo de lo previsto, en este caso la madera es un material difícil de trabajar con respecto a la exactitud de las perforaciones y las medidas, que en este caso fue un factor importante para retrasarnos en el proceso.

Las mediciones de una madera con respecto de la otra variaban mucho al medir y trazar no siempre se podía hacer exactamente, por ello es que nuestro material tiene muchas perforaciones, cada una es prácticamente por milésimas de error por eso es que no son tan céntricas las medidas.

La tarea que habíamos hecho anteriormente nos sirvió para entender la comunicación entre ros y los sistemas periféricos que en este caso nos ayudó demasiado ventaja con esa problemática aunque el sistema operativo mediante máquina virtual nos causa demasiados problemas de comunicación, que solo los podemos resolver reiniciando la máquina virtual o en su defecto reiniciar la computadora y ese es un problema que hasta el momento no se ha podido resolver.

La parte de interfaz nos llevó dos tarjetas freescale dañadas ya que las conexiones que realizamos no fueron las debidas por confiarnos, teníamos poco tiempo para entregar el producto final.

Los dos cortos circuitos que tuvo la tarjeta fue por un aparente crecimiento de corriente en la fuente externa, lo cual se decide medir constantemente la corriente eléctrica con el multímetro para monitorear constantemente la corriente

pico del sistema, se va utilizar de manera de protección y para monitorear el sistema si la corriente excede más de 2 amperes puede que la fuente externa quemara los motores o en su defecto dañe la tarjeta freescale.

Ana Zepeda

Para llevar a cabo este proyecto tuvimos que utilizar todos los conocimientos que habíamos adquirido anteriormente, por ejemplo la asignatura de metrología la utilizamos a la hora de tomar las medidas en la madera para poder acomodar los cojinetes e integrar los tubos aceros para crear un eje del robot, sistemas CAM y CAE fue otra asignatura utilizada, al realizar el diseño y posteriormente el análisis estático de la estructura para observar las posibles rupturas en el material que pudiera sufrir la estructura, aquí es donde tuvimos una de las principales fallas en la estructura.

Al realizar el análisis estático escogimos como material una aleación de metal para las bases del brazo, al realizar la estructura física utilizamos tablas de madera con pesos y medidas desiguales por lo cual los motores no fueron capaces de mover los espárragos correspondientes en cada eje.

Por lo tanto la mala implementación del material hizo que nuestro brazo robótico, no cumpliera con los parámetros establecidos por el profesor.

Hablando en términos de programación no hubo realmente problemas con la comunicación entre ROS y la freescale, sin embargo por un cálculo erróneo de corriente y la colocación de los drivers de manera invertida realizamos un puenteo que terminó por quemar el regulador de voltaje de los freescale haciendo que se sobrecalentara solo al conectarla a la PC, para resolver este problema realizamos una fase de potencia con un multímetro que nos permitiera observar y controlar la corriente que recibía la tarjeta y el CNC shield.

Anexos

Anexo 1

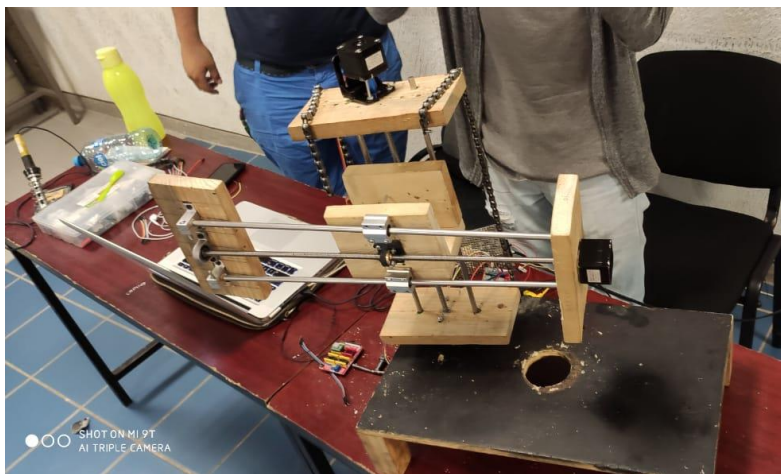


Figura 1.0 Estructura física del robot cilíndrico vista frontal.

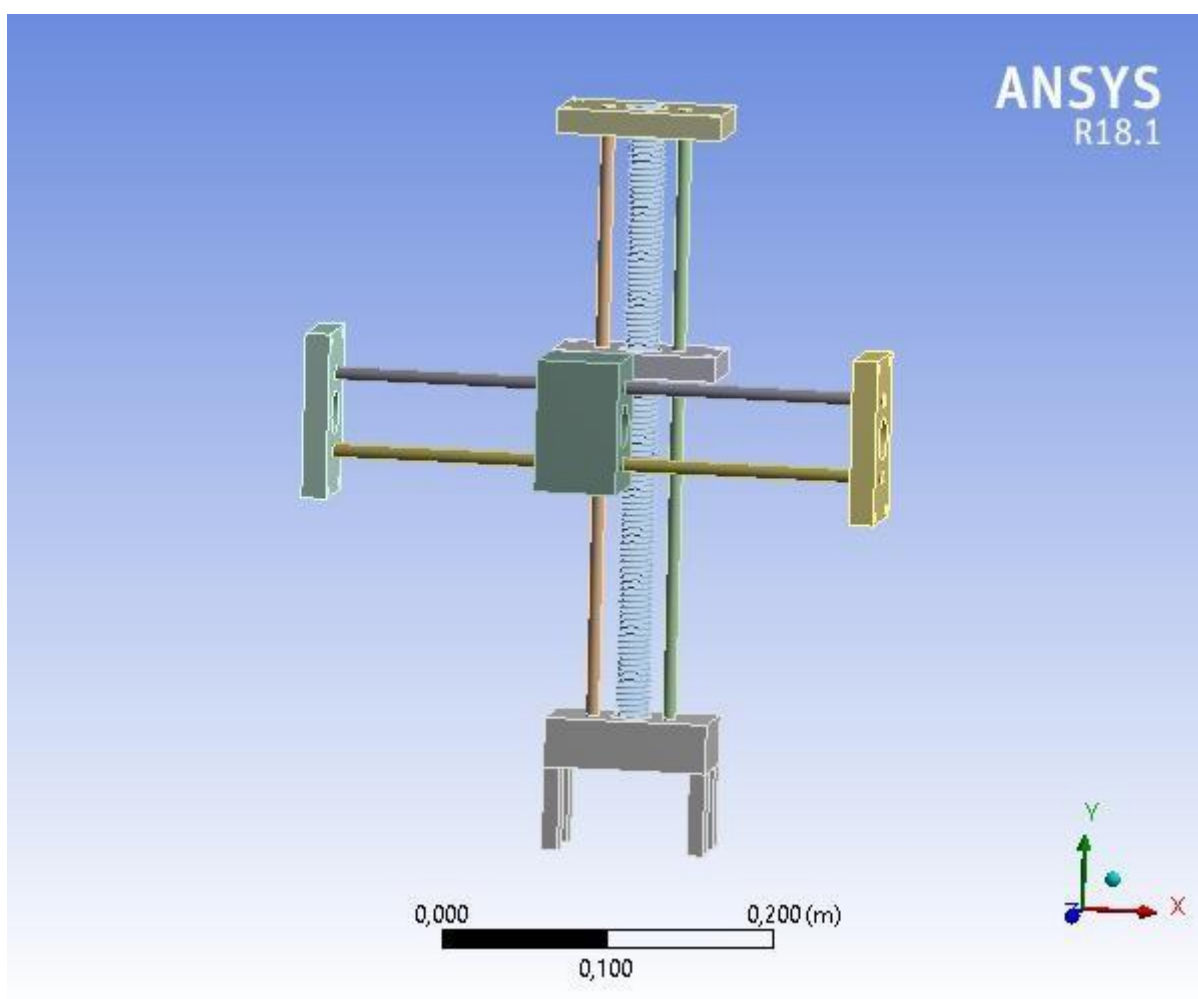


Figura 1.1 Estructura física del robot cilíndrico vista lateral derecha.

Anexo 2



First Saved	Wednesday, June 12, 2019
Last Saved	Wednesday, June 12, 2019
Product Version	18.1 Release
Save Project Before Solution	No
Save Project After Solution	No



Contents

- [Units](#)
- [Model \(B4\)](#)
 - [Geometry](#)
 - [Parts](#)
 - [Coordinate Systems](#)
 - [Connections](#)
 - [Contacts](#)
 - [Contact Regions](#)
 - [Mesh](#)
 - [Static Structural \(B5\)](#)
 - [Analysis Settings](#)
 - [Loads](#)
 - [Solution \(B6\)](#)
 - [Solution Information](#)
 - [Total Deformation](#)
- [Material Data](#)
 - [Structural Steel](#)

Report Not Finalized

Not all objects described below are in a finalized state. As a result, data may be incomplete, obsolete or in error. View first state problem. To finalize this report, edit objects as needed and solve the analyses.

Units

TABLE 1

Unit System	Metric (m, kg, N, s, V, A) Degrees rad/s Celsius
Angle	Degrees
Rotational Velocity	rad/s
Temperature	Celsius

Model (B4)

Geometry

TABLE 2

Object Name	Geometry
State	Fully Defined
Definition	
Source	C:\Users\PIIM\Documents\ROBOT\Ensamblaje final.IGS
Type	Iges
Length Unit	Meters
Element Control	Program Controlled
Display Style	Body Color
Length Y	Bounding Box 0,45 m
Length Z	0,051 m
Properties	
Volume	7,8225e-004 m ³
Mass	6,1407 kg
Scale Factor Value	1,
Statistics	
Bodies	12
Active Bodies	11
Nodes	28927
Elements	11127
Mesh Metric	None
Basic Geometry Options	
Solid Bodies	Yes
Surface Bodies	Yes
Line Bodies	No
Parameters	Independent
Parameter Key	ANS;DS
Attributes	No
Named Selections	No
Material Properties	No
Advanced Geometry Options	
Use Associativity	Yes
Coordinate Systems	No
Reader Mode Saves Updated File	No
Use Instances	Yes
Smart CAD Update	Yes
Compare Parts On Update	No
Attach File Via Temp File	Yes
Temporary Directory	C:\Users\PIIM\AppData\Roaming\Ansys\v181
Analysis Type	3-D
Mixed Import Resolution	None
Decompose Disjoint Geometry	Yes
Enclosure and Symmetry Processing	Yes

TABLE 4
Model (B4) > Geometry > Parts

Object Name	<i>Part 12</i>
State	Meshed
Graphics Properties	
Visible	Yes
Transparency	1
Definition	
Suppressed	No
Stiffness Behavior	Flexible
Coordinate System	Default Coordinate System
Reference Temperature	By Environment
Behavior	None
Material	
Assignment	Structural Steel
Nonlinear Effects	Yes
Thermal Strain Effects	Yes
Bounding Box	
Length X	1,e-001 m
Length Y	1,5e-002 m
Length Z	5,e-002 m
Properties	
Volume	6,4714e-005 m ³
Mass	0,50801 kg
Centroid X	-5,7042e-002 m
Centroid Y	0,22442 m
Centroid Z	0,30007 m
Moment of Inertia Ip1	1,261e-004 kg·m ²
Moment of Inertia Ip2	4,7268e-004 kg·m ²
Moment of Inertia Ip3	5,7971e-004 kg·m ²
Statistics	
Nodes	1412
Elements	204
Mesh Metric	None

Length X	1,e-001 m
Length Y	1,5e-002 m
Length Z	5,e-002 m
Properties	
Volume	6,4714e-005 m ³
Mass	0,50801 kg
Centroid X	-5,7042e-002 m
Centroid Y	0,22442 m
Centroid Z	0,30007 m
Moment of Inertia Ip1	1,261e-004 kg·m ²
Moment of Inertia Ip2	4,7268e-004 kg·m ²
Moment of Inertia Ip3	5,7971e-004 kg·m ²
Statistics	
Nodes	1412
Elements	204
Mesh Metric	None

Coordinate Systems

TABLE 5
Model (B4) > Coordinate Systems > Coordinate System

Object Name	<i>Global Coordinate System</i>
State	Fully Defined
Definition	
Type	Cartesian
Coordinate System ID	0,
Origin	
Origin X	0, m
Origin Y	0, m
Origin Z	0, m
Directional Vectors	
X Axis Data	[1, 0, 0,]
Y Axis Data	[0, 1, 0,]
Z Axis Data	[0, 0, 1,]

Connections

Object Name	<i>Connections</i>
State	Fully Defined
Auto Detection	
Generate Automatic Connection On Refresh	Yes
Transparency	
Enabled	Yes

TABLE 6
Model (B4) > Connections

Object Name	<i>Contacts</i>
State	Fully Defined
Definition	
Connection Type	Contact

TABLE 7
Model (B4) > Connections > Contacts

Scope	
Scoping Method	Geometry Selection
Geometry	All Bodies
Auto Detection	
Tolerance Type	Slider
Tolerance Slider	0,
Tolerance Value	1,4475e-003 m
Use Range	No
Face/Face	Yes
Face Overlap Tolerance	Off
Cylindrical Faces	Include
Face/Edge	No
Edge/Edge	No
Priority	Include All
Group By	Bodies
Search Across	Bodies
Statistics	
Connections	17
Active Conn	14
ections	

TABLE 8**Model (B4) > Connections > Contacts > Contact
Regions**

Elastic Slip Tolerance	Program Controlled
Normal Stiffness	Program Controlled
Update Stiffness	Program Controlled
Pinball Region	Program Controlled
Geometric Modification	
Contact Geometry Correction	None
Target Geometry Correction	None

TABLE 9
Model (B4) > Connections > Contacts > Contact
Regions

Object Name	Contact Region 12	Contact Region 13	Contact Region 14	Contact Region 15	Contact Region 16	Contact Region 17
State	Fully Defined					
Scope						
Scoping Method	Geometry Selection					
Contact	2 Faces					
Target	2 Faces					
Contact Bodies	Part 5		Part 7		Part 8	
Target Bodies	Part 7	Part 8	Part 11	Part 12	Part 11	Part 12
Definition						
Type	Bonded					
Scope Mode	Automatic					
Behavior	Program Controlled					
Trim Contact	Program Controlled					
Trim Tolerance	1,4475e-003 m					
Suppressed	No					
Advanced						
Formulation	Program Controlled					
Detection Method	Program Controlled					
Penetration Tolerance	Program Controlled					
Elastic Slip Tolerance	Program Controlled					
Normal Stiffness	Program Controlled					
Update Stiffness	Program Controlled					
Pinball Region	Program Controlled					
Geometric Modification						
Contact Geometry Correction	None					
Target Geometry Correction	None					

Mesh

TABLE 10
Model (B4) > Mesh

Object Name	<i>Mesh</i>
State	Solved
Display	
Display Style	Body Color
Defaults	
Physics Preference	Mechanical
Relevance	0
Element Order	Program Controlled
Sizing	
Size Function	Adaptive
Relevance Center	Coarse
Element Size	Default
Initial Size Seed	Assembly
Transition	Fast
Span Angle Center	Coarse
Automatic Mesh Based Defeaturing	On
Defeature Size	Default
Minimum Edge Length	2,6787e-005 m
Quality	
Check Mesh Quality	Yes, Errors
Error Limits	Standard Mechanical
Target Quality	Default (0.050000)
Smoothing	Medium
Mesh Metric	None
Inflation	
Use Automatic Inflation	None
Inflation Option	Smooth Transition
Transition Ratio	0,272
Maximum Layers	5
Growth Rate	1,2
Inflation Algorithm	Pre
View Advanced Options	No
Advanced	
Number of CPUs for Parallel Part Meshing	Program Controlled
Straight Sided Elements	No
Number of Retries	4
Rigid Body Behavior	Dimensionally Reduced
Mesh Morphing	Disabled
Triangle Surface Mesher	Program Controlled
Topology Checking	No
Pinch Tolerance	Please Define
Generate Pinch on Refresh	No
Statistics	
Nodes	28927
Elements	11127

Static Structural (B5)

TABLE 12
Model (B4) > Static Structural (B5) > Analysis

Object Name	<i>Static Structural (B5)</i>
State	Solved
Definition	
Physics Type	Structural
Analysis Type	Static Structural
Solver Target	Mechanical APDL
Options	
Environment Temperature	22, °C
Generate Input Only	No

Settings

Object Name	<i>Analysis Settings</i>
State	Fully Defined
Step Controls	
Number Of Steps	1,
Current Step Number	1,
Step End Time	2,e-004 s
Auto Time Stepping	Program Controlled
Solver Controls	
Solver Type	Program Controlled
Weak Springs	Off
Solver Pivot Checking	Program Controlled
Large Deflection	Off
Inertia Relief	Off
Rotordynamics Controls	
Coriolis Effect	Off
Restart Controls	
Generate Restart Points	Program Controlled
Retain Files After Full Solve	No
Combined Restart Files	Program Controlled
Nonlinear Controls	
Newton-Raphson Option	Program Controlled
Force Convergence	Program Controlled
Moment Convergence	Program Controlled
Displacement Convergence	Program Controlled
Rotation Convergence	Program Controlled
Line Search	Program Controlled
Stabilization	Off
Output Controls	
Stress	Yes
Strain	Yes
Nodal Forces	No
Contact Miscellaneous	No
General Miscellaneous	No

Store Results At	All Time Points
Analysis Data Management	
Solver Files Directory	F:\Simulacion2_files\dp0\SYSMECH\
Future Analysis	None
Scratch Solver Files Directory	
Save MAPDL db	No
Delete Unneeded Files	Yes

(B4) > Analysis

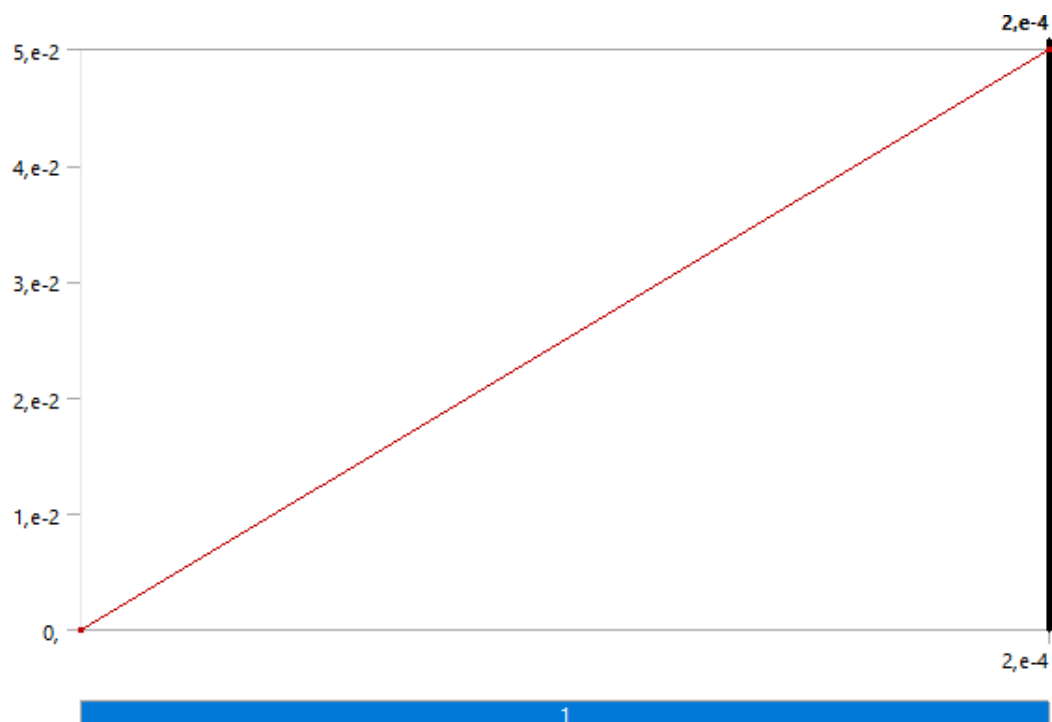
Nonlinear Solution	No
Solver Units	Active System
Solver Unit System	mks

TABLE 13
Model (B4) > Static Structural (B5) > Loads

Object Name	Fixed Support	Pressure
State	Fully Defined	
Scope		
Scoping Method	Geometry Selection	
Geometry	29 Faces	2 Faces
Definition		
Type	Fixed Support	Pressure
Suppressed	No	
Define By	Normal To	
Applied By	Surface Effect	
Magnitude	5,e-002 Pa (ramped)	

FIGURE 1

Model (B4) > Static Structural (B5)



Solution (B6)

TABLE 14
Model (B4) > Static Structural (B5) > Solution

Object Name	<i>Solution (B6)</i>
State	Solved
Adaptive Mesh Refinement	
Max Refinement Loops	1,
Refinement Depth	2,

Information	
Status	Done
MAPDL Elapsed Time	1 m 16 s
MAPDL Memory Used	181, MB
MAPDL Result File Size	12,75 MB
Post Processing	
Beam Section Results	No

TABLE 15
Model (B4) > Static Structural (B5) > Solution
(B6) > Solution Information

Object Name	<i>Solution Information</i>
State	Solved
Solution Information	
Solution Output	Solver Output
Newton-Raphson Residuals	0
Identify Element Violations	0
Update Interval	2,5 s
Display Points	All
FE Connection Visibility	
Activate Visibility	Yes
Display	All FE Connectors
Draw Connections Attached To	All Nodes
Line Color	Connection Type
Visible on Results	No
Line Thickness	Single
Display Type	Lines

TABLE 16
Model (B4) > Static Structural (B5) > Solution
(B6) > Results

FIGURE 2
Model (B4) > Static Structural (B5) > Solution

Object Name	<i>Total Deformation</i>
State	Solved
Scope	
Scoping Method	Geometry Selection
Geometry	All Bodies
Definition	
Type	Total Deformation
By	Time
Display Time	Last
Calculate Time History	Yes
Identifier	
Suppressed	No
Results	
Minimum	0, m
Maximum	1,6649e-016 m
Minimum Occurs On	Part 1
Maximum Occurs On	Part 3
Information	
Time	2,e-004 s
Load Step	1

(B6) > Total Deformation

TABLE 17
Model (B4) > Static Structural (B5) > Solution
(B6) > Total Deformation

Time [s]	Minimum [m]	Maximum [m]
2,e-004	0,	1,6649e-016

FIGURE 3
Model (B4) > Static Structural (B5) > Solution
(B6) > Total Deformation > Figure

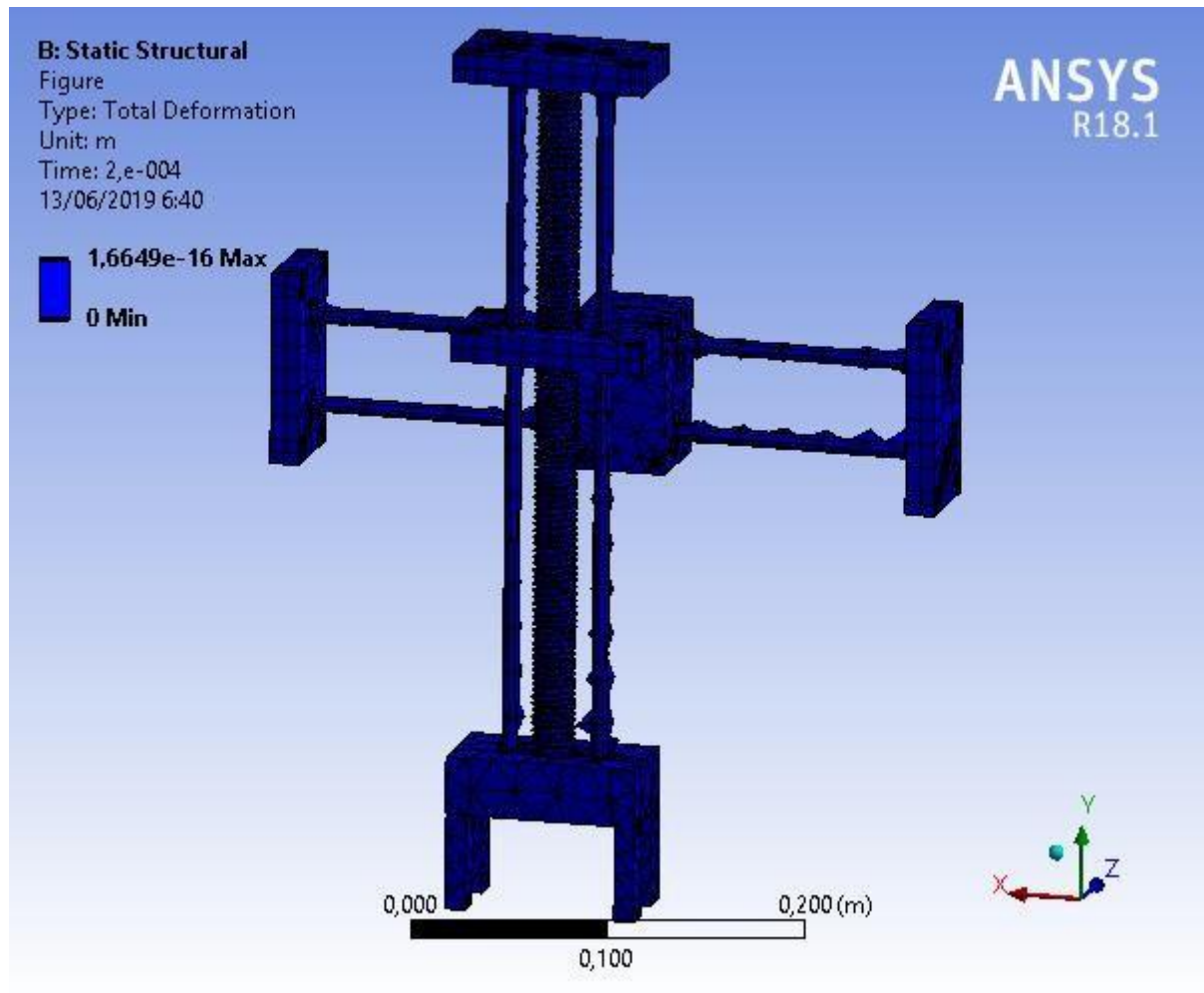


FIGURE 4
Model (B4) > Static Structural (B5) > Solution
(B6) > Total Deformation > Figure 2

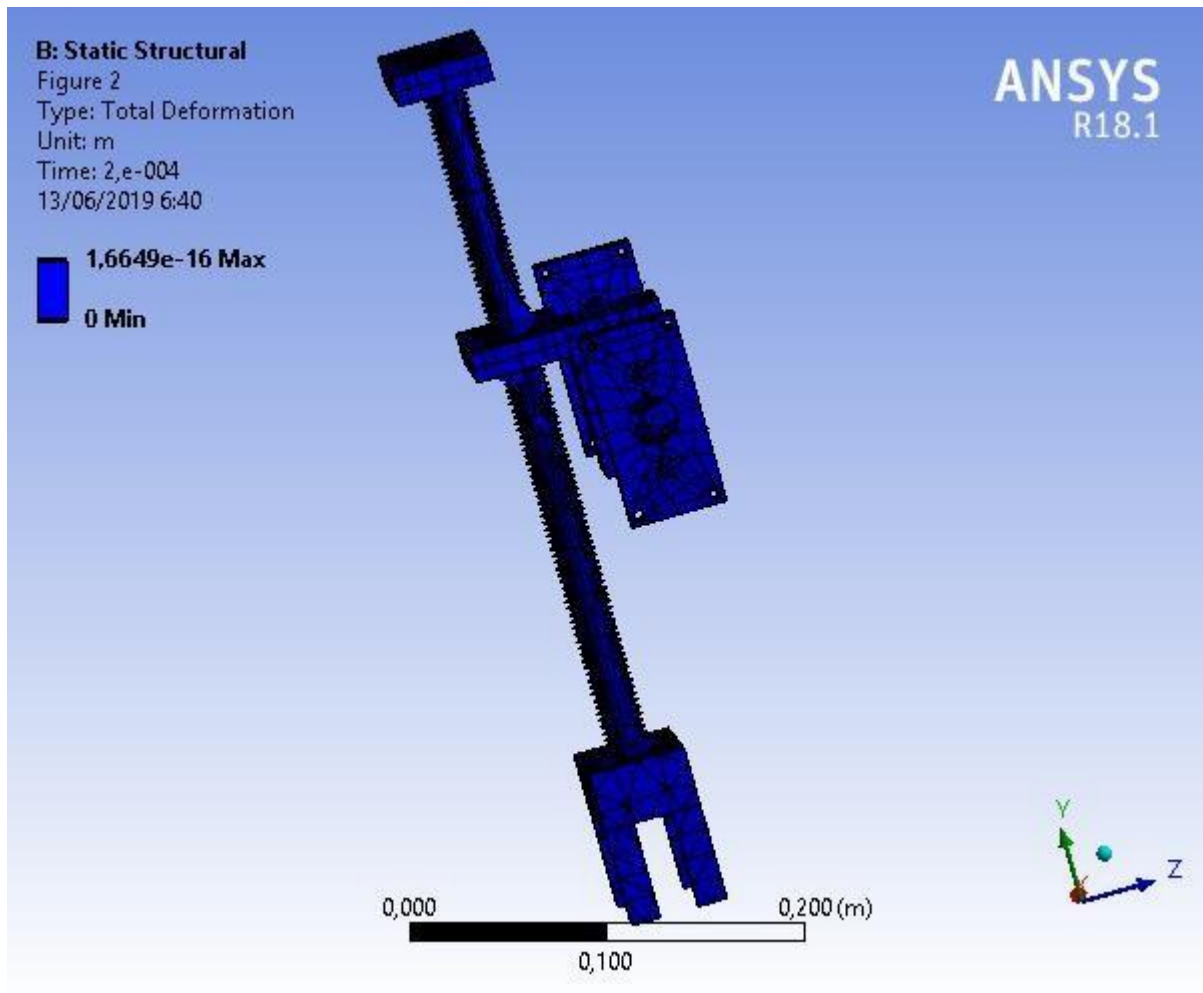
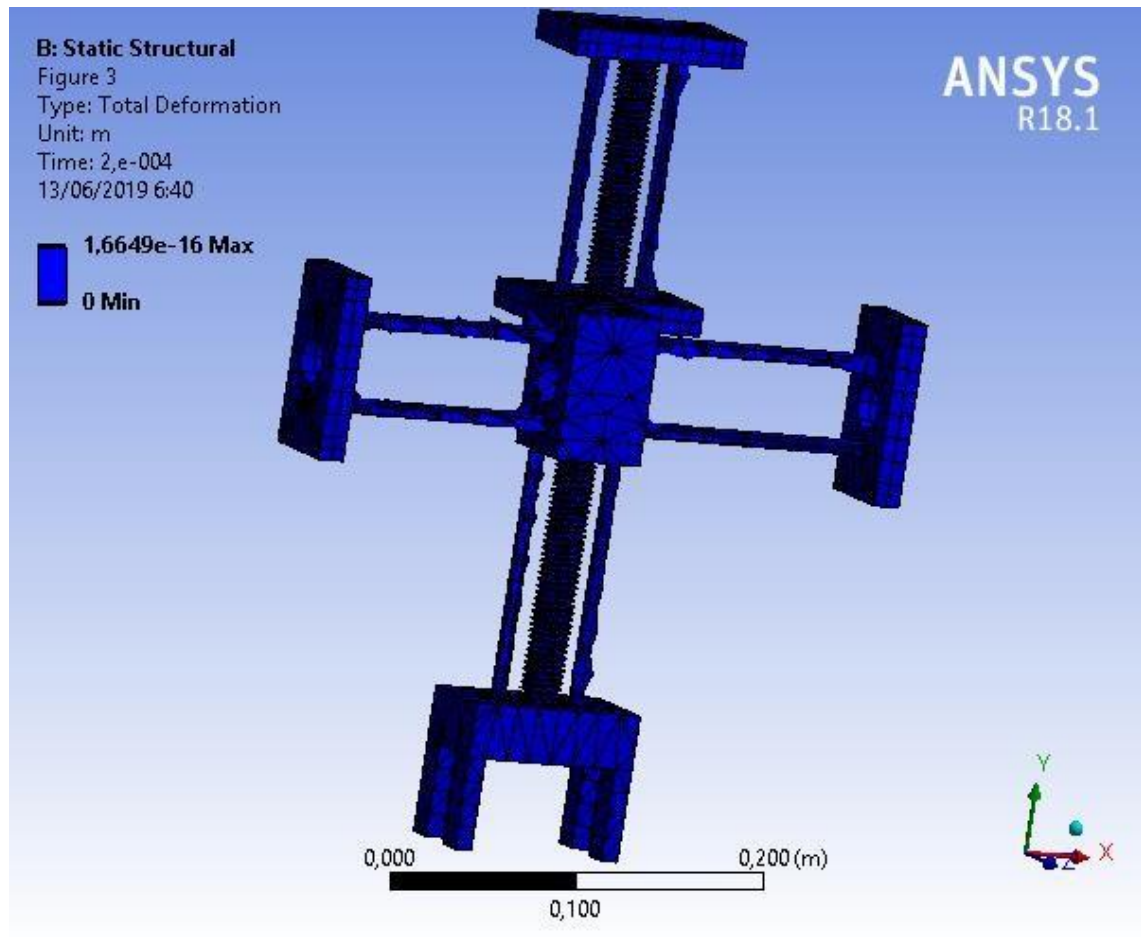


FIGURE 5
Model (B4) > Static Structural (B5) > Solution
(B6) > Total Deformation > Figure 3



Structural Steel

TABLE 18
Structural Steel > Constants

Density	7850, kg m ⁻³
Isotropic Secant Coefficient of Thermal Expansion	1,2e-005 C ⁻¹
Specific Heat	434, J kg ⁻¹ C ⁻¹
Isotropic Thermal Conductivity	60,5 W m ⁻¹ C ⁻¹
Isotropic Resistivity	1,7e-007 ohm m

TABLE 19
Structural Steel > Appearance

Red	Green	Blue
132,	139,	179,

TABLE 20
Structural Steel > Compressive Ultimate Strength

Compressive Ultimate Strength Pa
0,

TABLE 21
Structural Steel > Compressive Yield Strength

Compressive Yield Strength Pa
2,5e+008

TABLE 22
Structural Steel > Tensile Yield Strength

Tensile Yield Strength Pa
2,5e+008

TABLE 23
Structural Steel > Tensile Ultimate Strength

Tensile Ultimate Strength Pa
4,6e+008

TABLE 24
Structural Steel > Isotropic Secant Coefficient of Thermal Expansion

Zero-Thermal-Strain Reference Temperature C
22,

TABLE 25
Structural Steel > Alternating Stress Mean Stress

Alternating Stress Pa	Cycles	Mean Stress Pa
3,999e+009	10,	0,
2,827e+009	20,	0,
1,896e+009	50,	0,
1,413e+009	100,	0,
1,069e+009	200,	0,
4,41e+008	2000,	0,
2,62e+008	10000	0,
2,14e+008	20000	0,
1,38e+008	1,e+005	0,
1,14e+008	2,e+005	0,
8,62e+007	1,e+006	0,

TABLE 26
Structural Steel > Strain-Life Parameters

Strength Coefficient Pa	Strength Exponent	Ductility Coefficient	Ductility Exponent	Cyclic Strength Coefficient Pa	Cyclic Strain Hardening Exponent
9,2e+008	-0,106	0,213	-0,47	1,e+009	0,2

TABLE 27
Structural Steel > Isotropic Elasticity

Temperature C	Young's Modulus Pa	Poisson's Ratio	Bulk Modulus Pa	Shear Modulus Pa
	2,e+011	0,3	1,6667e+011	7,6923e+010

TABLE 28
Structural Steel > Isotropic Relative Permeability

Relative Permeability
10000

Bibliografía

Antonio, J. (16 de Marzo de 2014). *Slideshare*. Obtenido de

<https://es.slideshare.net/paviruchi/tema-2-estructura-mecanica-de-un-robot>

Romo, E. (06 de Septiembre de 2015). Obtenido de PREZI: <https://prezi.com/chuyzghoett2/robots-de-coordenadas-cilindricas/>

U de Santiago Virtual. (s.f.). Obtenido de Moodle:

<http://www.udesantiagoovirtual.cl/moodle2/mod/book/view.php?id=24911&chapterid=225>