ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ENERGÍA Y MECÁNICA
CARRERA DE INGENIERÍA MECATRÓNICA
TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MECATRÓNICO

TEMA: IMPLEMENTACIÓN DE UN ALGORITMO FLEXIBLE EN EL ROBOT MITSUBISHI MELFA RV-2SDB QUE PERMITA MEJORAR LA EFICIENCIA EN LA PROGRAMACIÓN UTILIZANDO UNIDADES DE MEDIDA INERCIAL

AUTORES:

PICHUCHO CASTELLANO, JEFFERSON PAÚL

SAMPEDRO GÓMEZ, ANDRÉS MARCELO

DIRECTOR:

ING. MENDOZA CHIPANTASI, DARÍO JOSÉ

LATACUNGA
2020

AGENDA

- Objetivos
- Metodología
- Marco Teórico
- Diseño mecánico
- Diseño electrónico
- Diseño computacional
- Algoritmos Flexibles
- Interfaz Gráfica
- Pruebas y resultados
- Conclusiones y recomendaciones
- Video

IMPLEMENTACIÓN DE UN ALGORITMO FLEXIBLE EN EL ROBOT MITSUBISHI MELFA RV-2SDB QUE PERMITA MEJORAR LA EFICIENCIA EN LA PROGRAMACIÓN UTILIZANDO UNIDADES DE MEDIDA INERCIAL

.

Objetivos específicos

Recopilar información sobre robótica industrial, telerrobótica, sensores que manejen unidades de medida inercial, y lenguaje de programación aplicado a brazos robóticos.

Adquirir y procesar por computador las señales de medida inercial procedentes del sensor utilizando software libre.

Establecer un enlace entre los componentes del sistema: sensores, controlador y ordenador manejando distintas formas de comunicación.

Diseñar e implementar el algoritmo flexible de movimiento en el robot mediante lenguaje estructurado u orientado a objetos. .

Comprobar la eficiencia en la programación al utilizar unidades de medida inercial, mediante pruebas de funcionamiento del algoritmo en el brazo robótico.

Metodología

Método de recopilación de la información

- Recopilar la información para documentar un fenómeno.

Método experimental

- Método empírico que pone a prueba el algoritmo flexible.

Método hipotético-deductivo

- Observación del fenómeno para plantear la hipótesis

Anatomía y Biomecánica del Brazo Humano

- Hombro
- Codo
- Muñeca

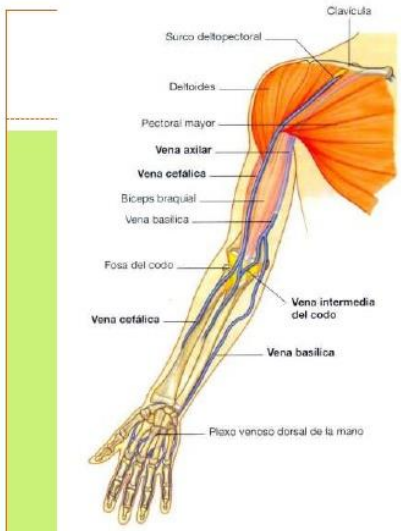


Fig. 7.18 Venas en la fascia superficial de la extremidad superior. La zona de la fosa del codo se muestra en amarillo.

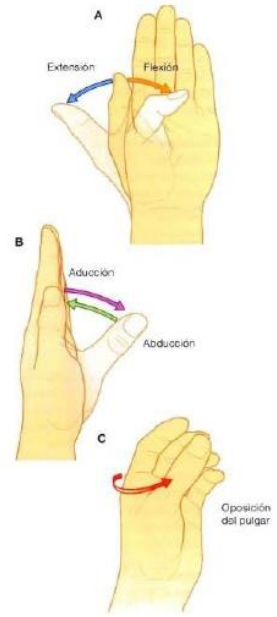


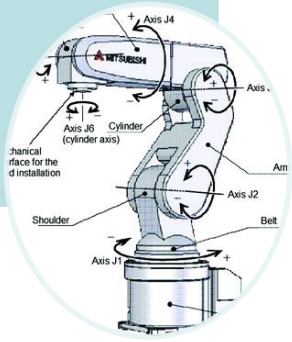
Fig. 7.19 Movimientos del pulgar.

- Robótica
- Robot Industrial Manipulador

Definición de Robot

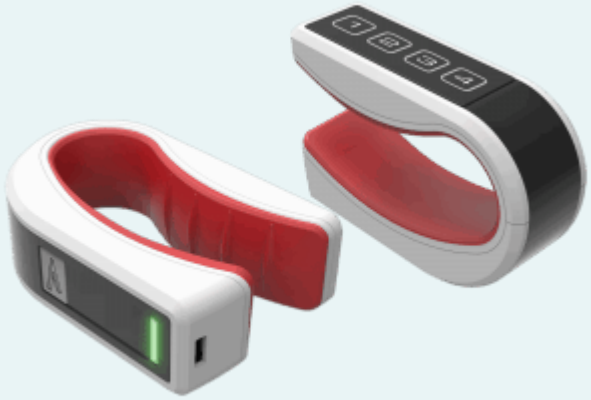
- Teleoperación
- Telerrobótica
- Telemanipulación

Brazo Robótico Mitsubishi MELFA RV-2SDB



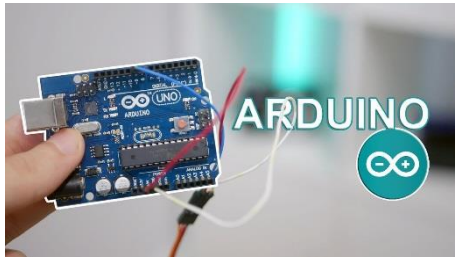
Sistema de Captura de Movimiento Inercial

- Sensores de Medición Inercial
- Giroscopio
- Sensor Magnetómetro
- Acelerómetro

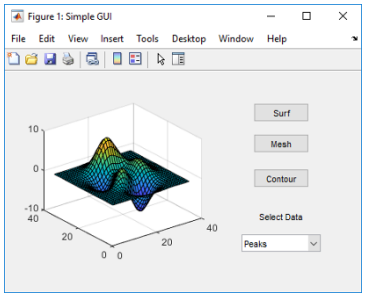


Herramientas Computacionales

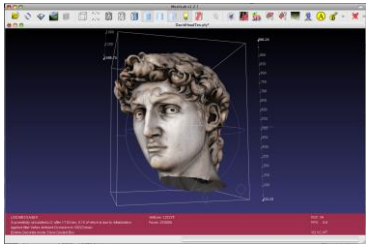
Arduino



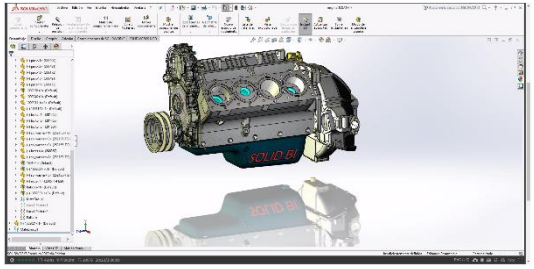
Matlab



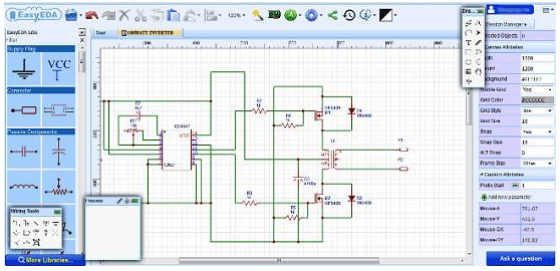
MeshLab



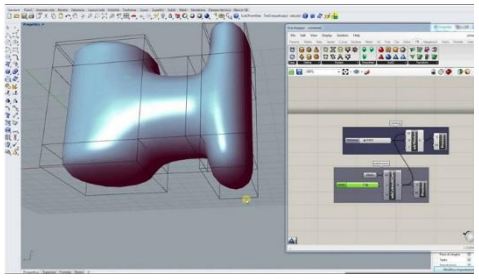
SolidWorks



EasyEDA

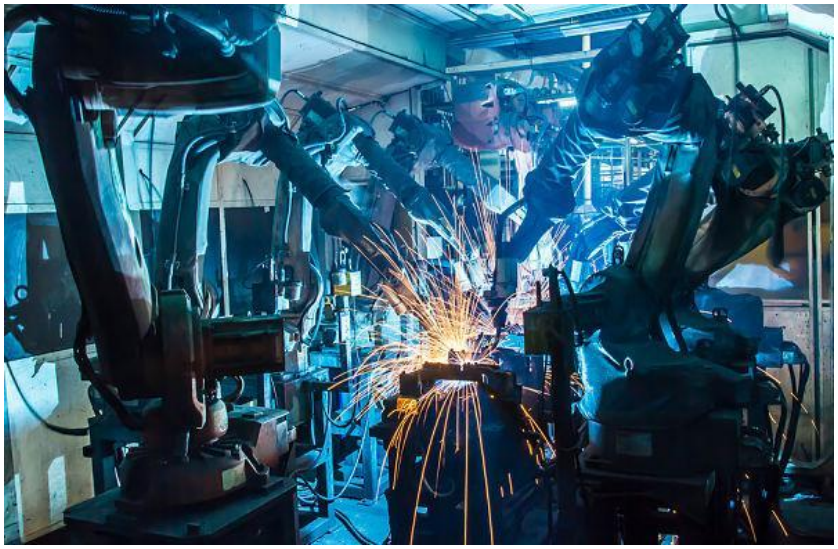


Rhinoceros 3D



Aplicaciones de los Robots

Aplicación de materiales pintura



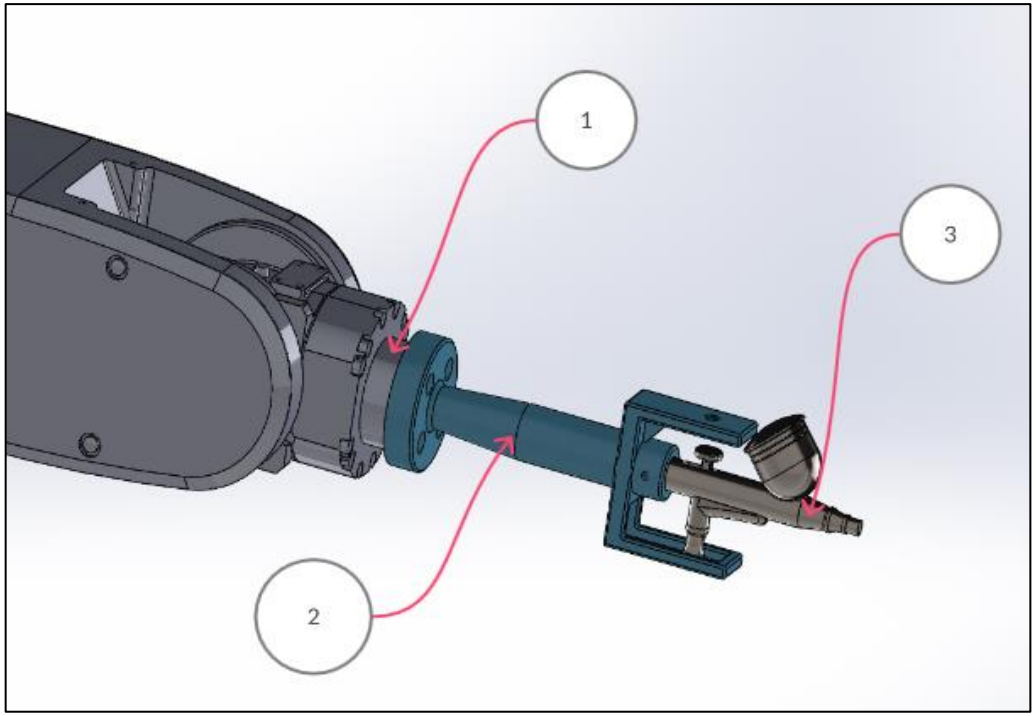
Requerimientos del Operador

- Una interfaz para interactuar con el brazo robótico.
- Programación sencilla.
- Los dispositivos electrónicos a utilizar deben ser fáciles de implementar y con estructura robusta.
- Debe ser de un bajo costo.

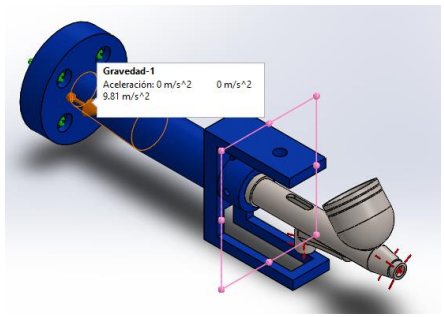
Requerimientos Técnicos

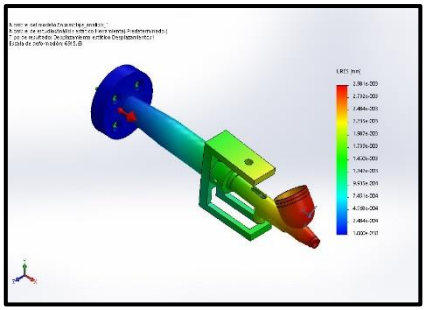
- El algoritmo debe poseer distintos modos de operación como tele operación y programación.
- Sensores de posiciones angulares fáciles de implementar.
- Sensores con conexión inalámbrica y con autonomía de carga eléctrica.
- Placa de circuito impreso.
- Estructura, materiales y dimensiones para la base de la herramienta de trabajo.
- Conexión sencilla.
- Sistema de bajo costo

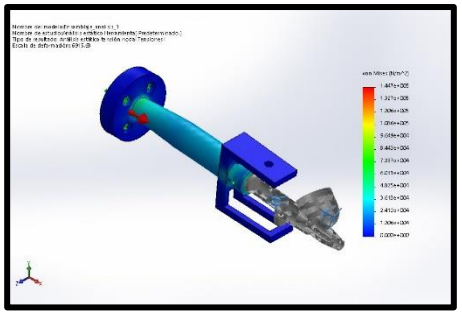
- **Diseño e Implementación del Sistema Mecánico**



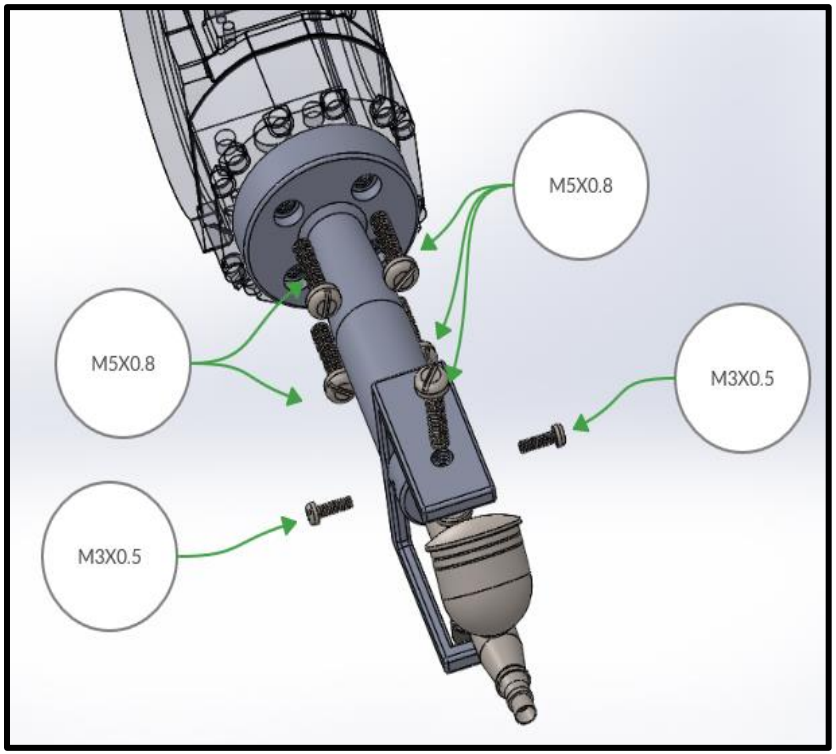
- **Análisis CAE del Portaherramientas**



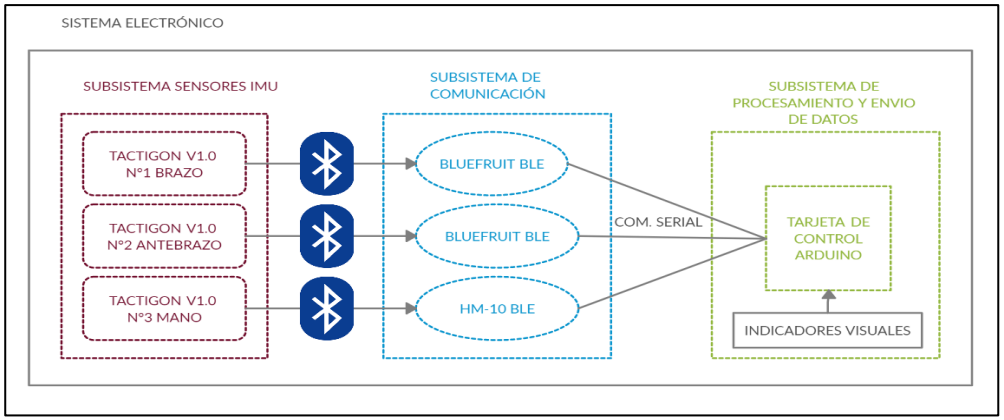




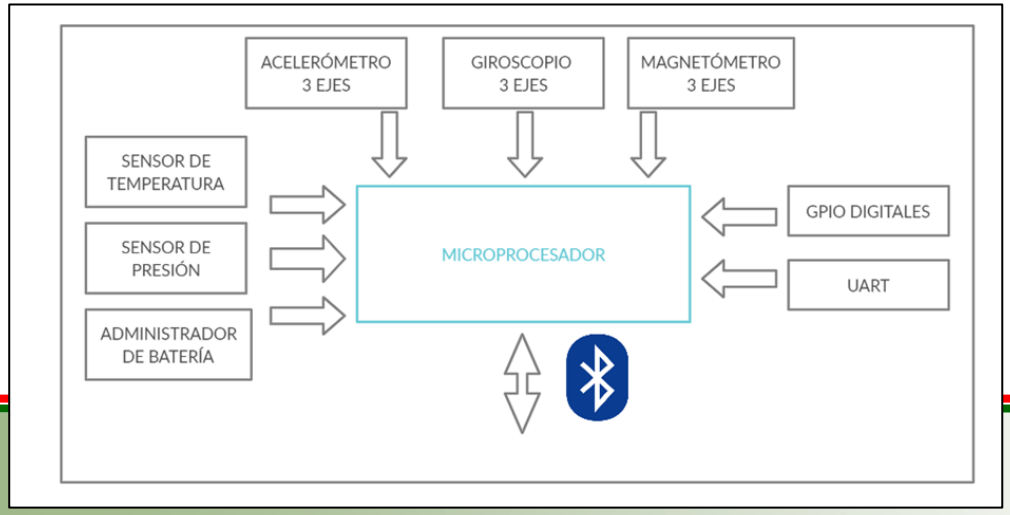
- **Implementación del Sistema Mecánico**



Esquema del sistema electrónico

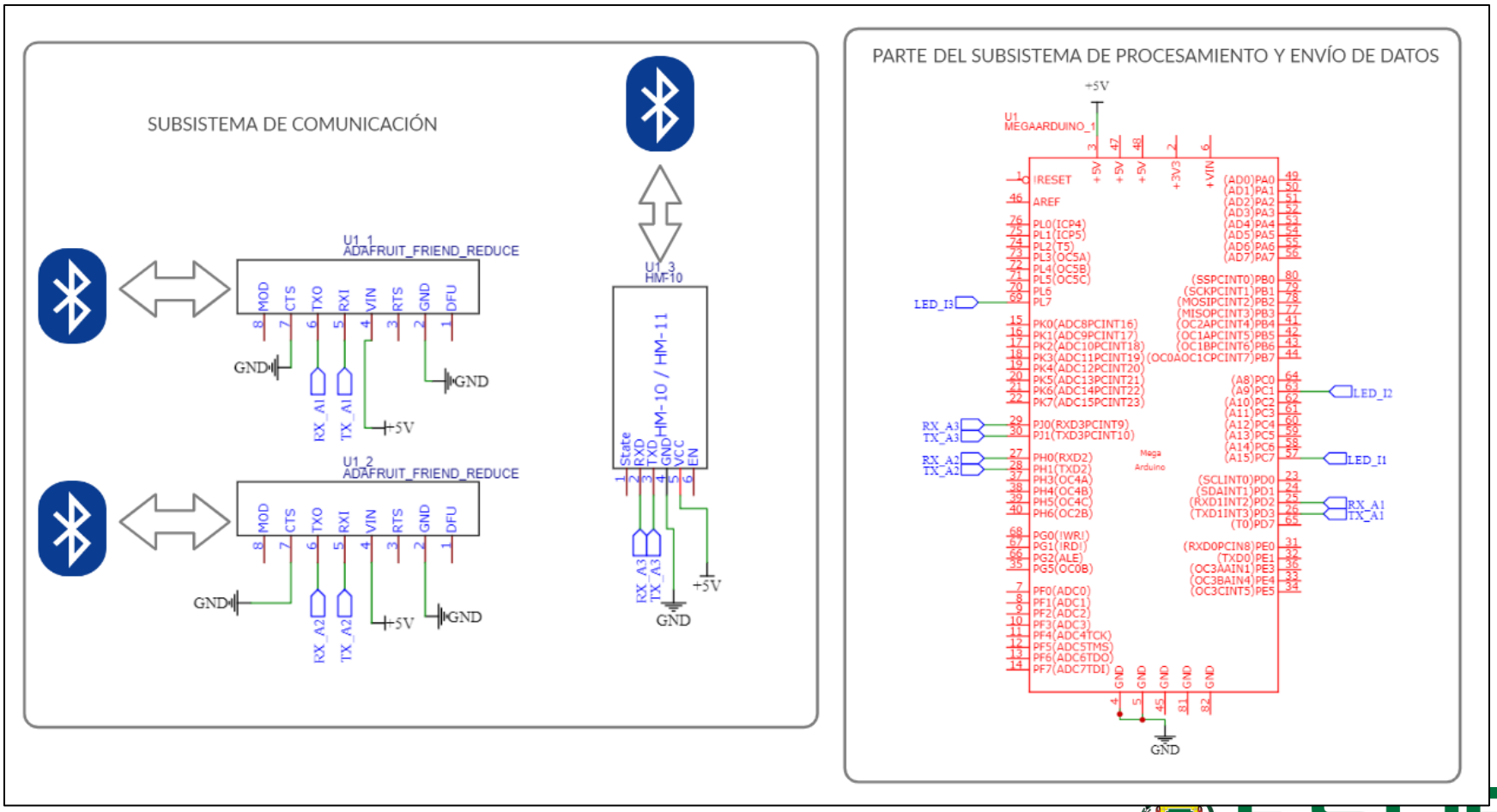


Arquitectura del sensor Tactigon ONE V1.0

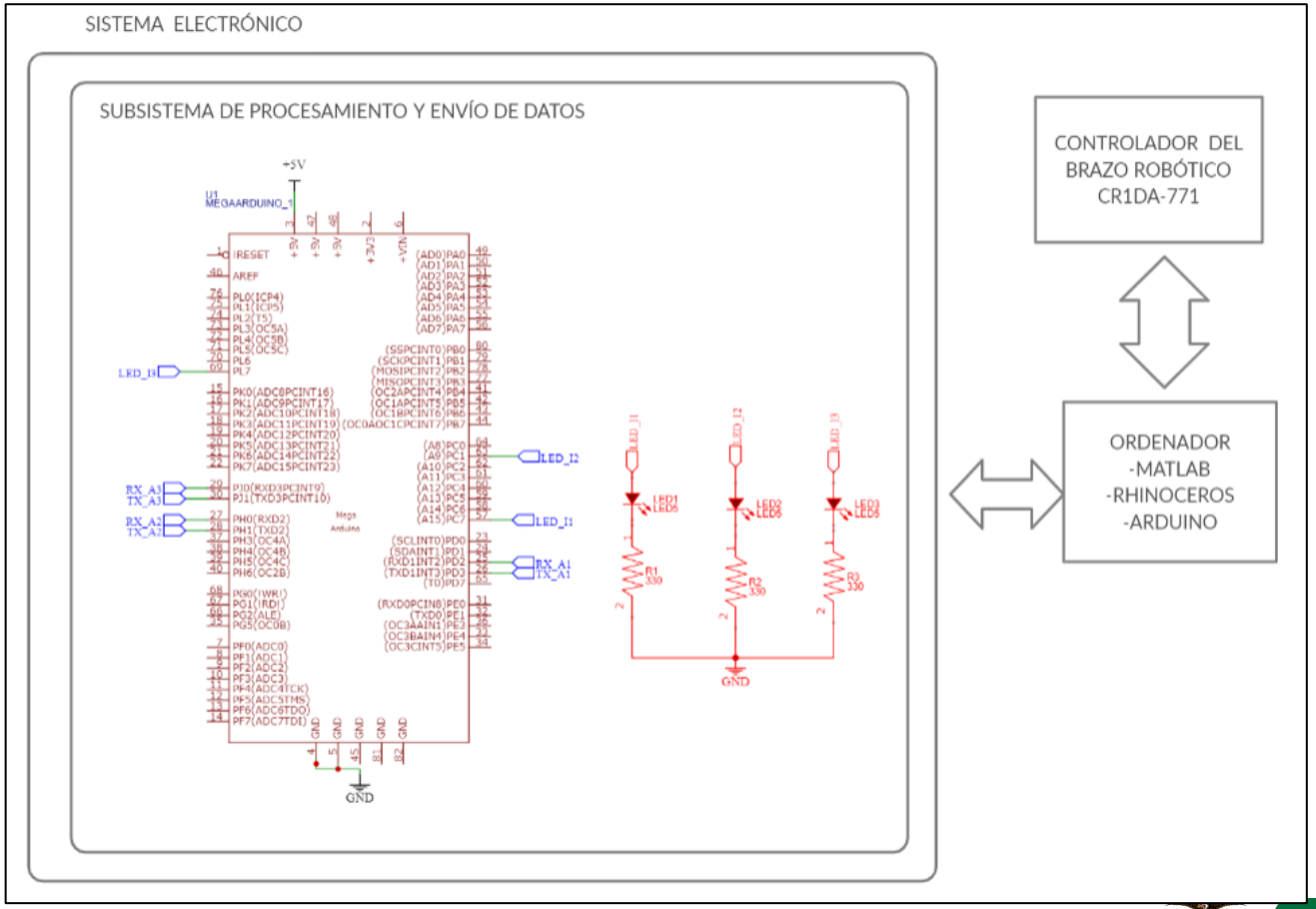


Algoritmo de procesamiento Tactigon ONE V1.0

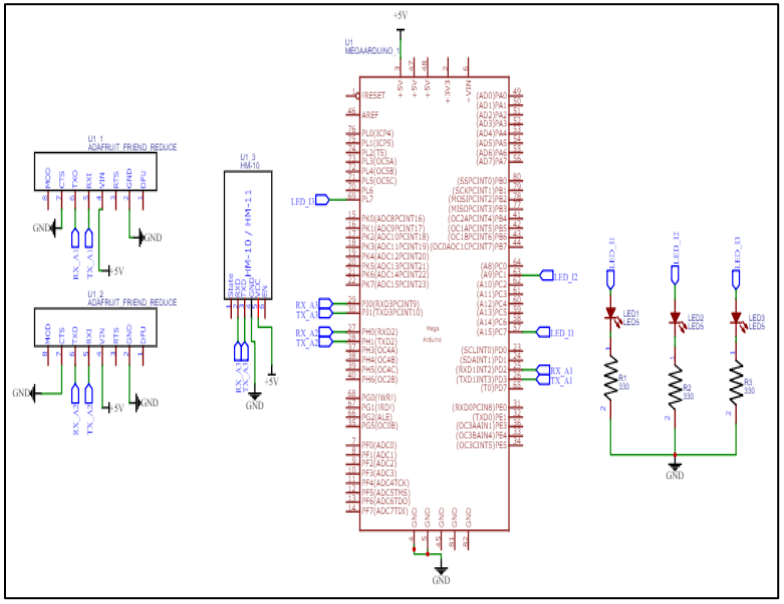
Subsistema de Comunicación

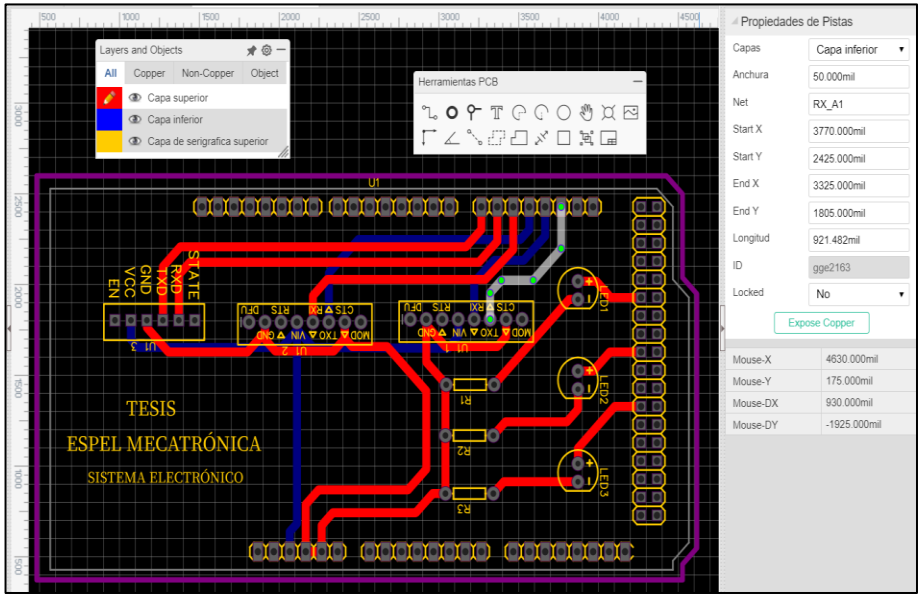


Subsistema de procesamiento y envío de datos

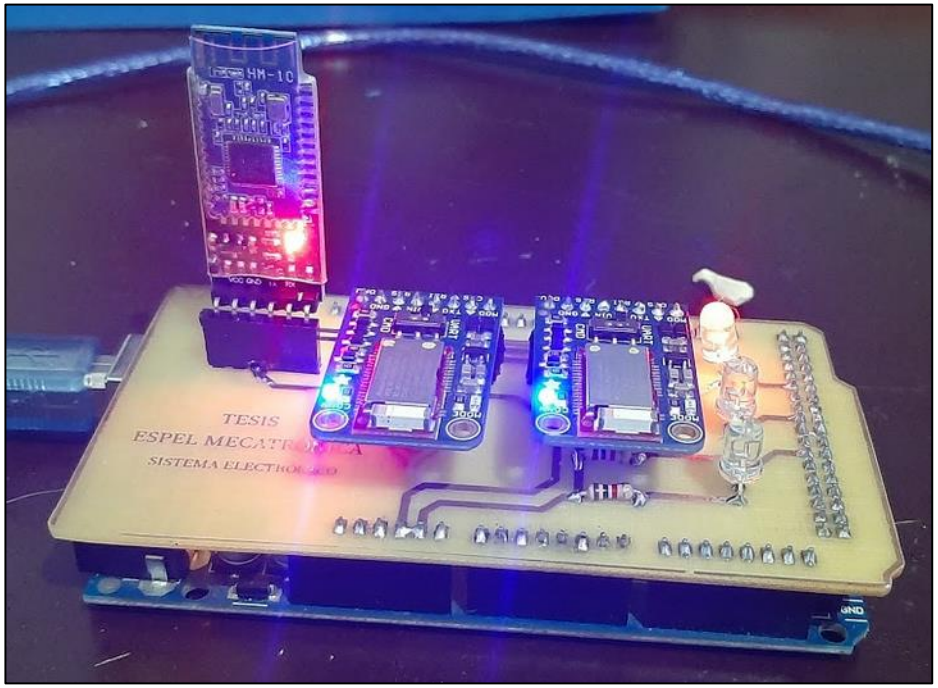


Diseño de la PCB

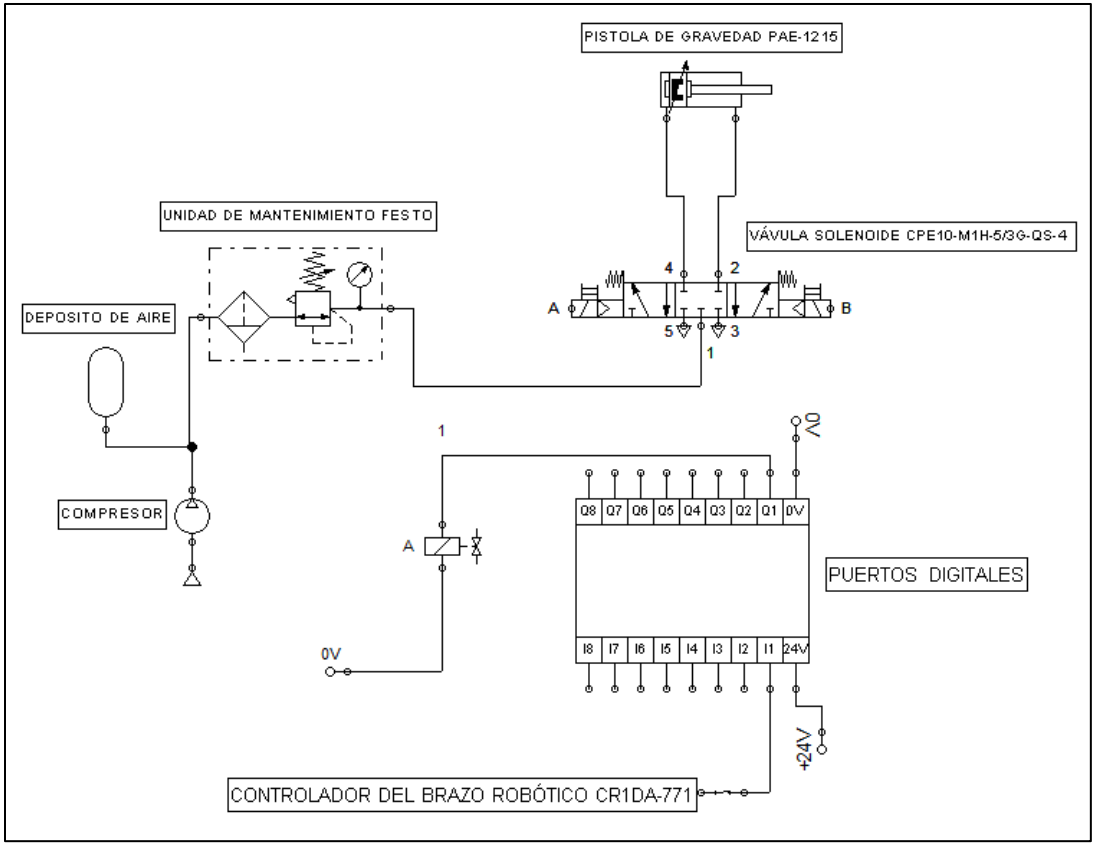




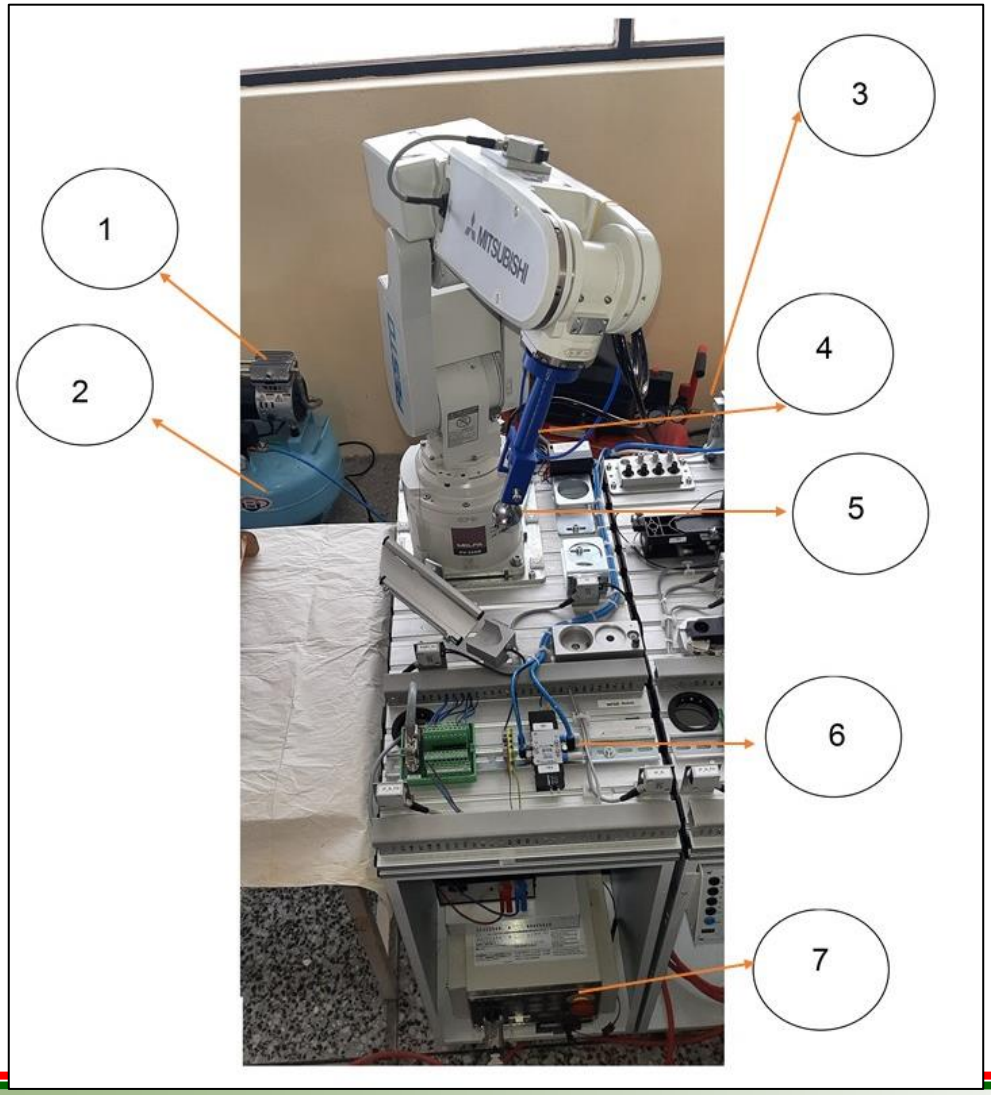
Implementación del sistema electrónico



Esquema del Sistema Neumático

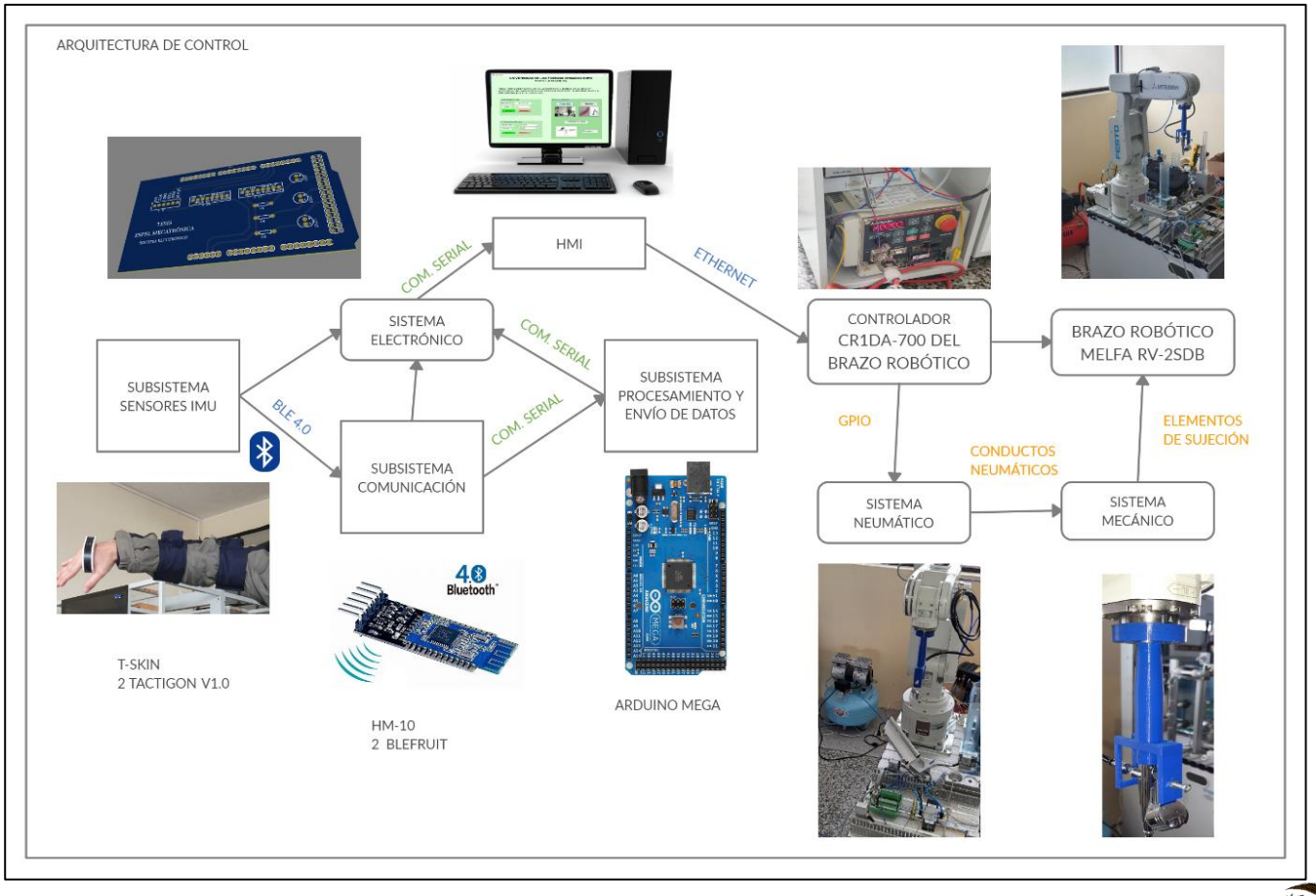


Implementación del Sistema Neumático



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Arquitectura de Control



Parámetros Denavit Hartenberg

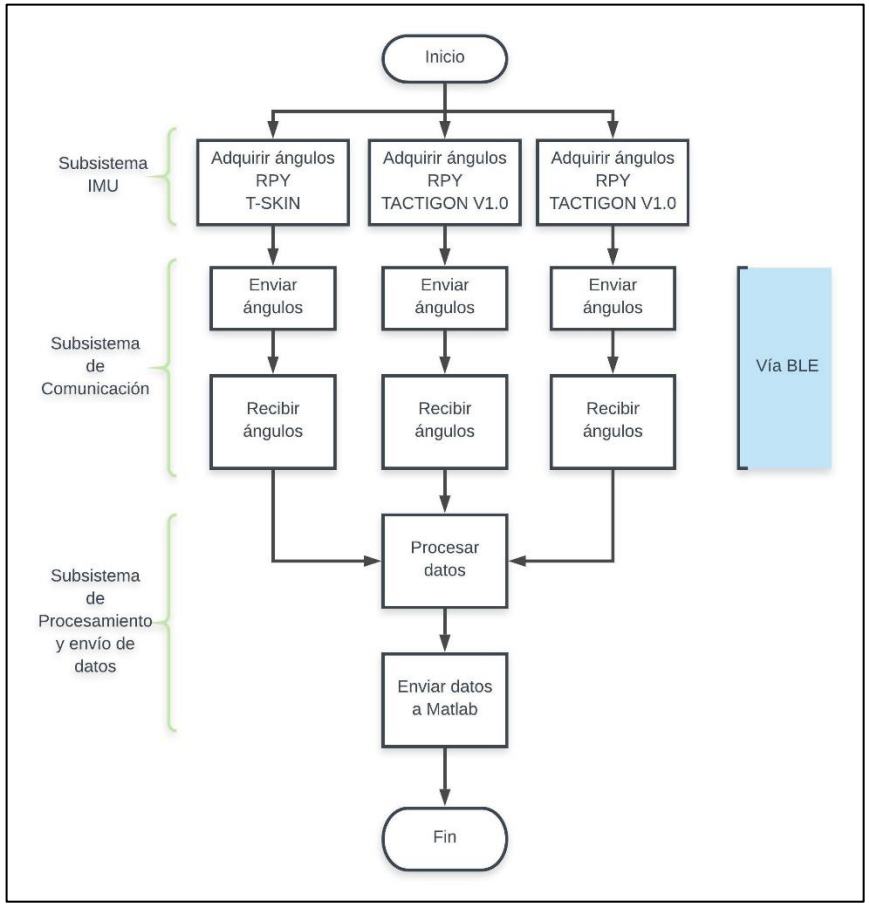
θ	$q(1)$	$q(2) - \frac{\pi}{2}$	$q(3)$	$q(4)$	$q(5)$	$q(6)$
D	0.295	0	0	0.27	0	0.07
a	0	0.23	0.05	0	0	0
∝	-pi/2	0	-pi/2	pi/2	-pi/2	0

DH del Brazo Humano

Link (1)	0	0	0	-pi/2
Link (2)	0	0	0.34	-pi/2
Link (3)	0	0	0	Pi/2
Link (4)	0	0	0.33	Pi/2

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo de Control del Sistema Electrónico

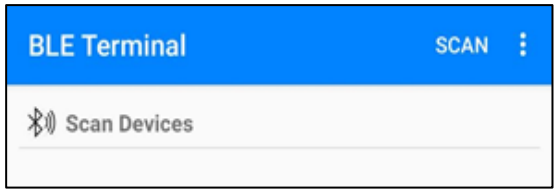


Programación del Subsistema IMU

```
3 #include <tactigon_led.h>
4 #include <tactigon_IMU.h>
5 #include <tactigon_BLE.h>
6 #include <tactigon_IO.h>
7
```

```
101     qData = qMeter.getQs();
102     roll=round(radToDeg(qData.roll));
103     yaw=round(radToDeg(qData.yaw));
104     pitch=round(radToDeg(qData.pitch));
```

Programación del Subsistema de Comunicación




```
#define TARGET_MAC {0xEB, 0x1B, 0xD2, 0xD7, 0x6B, 0x77};
#define TARGET_CHAR "6e400002-b5a3-f393-e0a9-e50e24dcca9e
 bleManager.InitRole(TACTIGON_BLE_CENTRAL);
 targetUUID.set(TARGET_CHAR);
 bleManager.setTarget(targetMAC, targetUUID);
 bleManager.writeToPeripheral((unsigned char *)bleBuff, strlen(bleBuff));
```

Características del módulo Adafruit Bluefruit #2

Envío de ángulos

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Programación del Subsistema de Procesamiento y Envío de Datos.

```
Serial1.begin(9600);
Serial2.begin(9600);
Serial3.begin(9600);
```

Inicializar puertos seriales

```
while (Serial1.available())
```

Bucle de un puerto serial activo

```
//sscanf permite receptar los datos de los módulos ble
match1 = sscanf(buff1, "u%dr%dy%d", &num1, &a11, &a21);
```

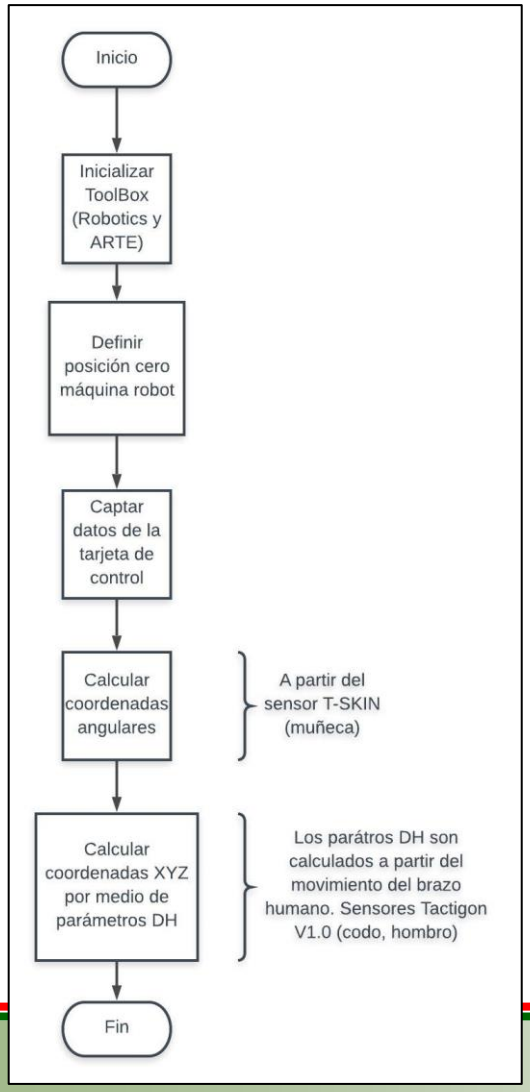
Recepción de ángulos

```
Serial.print(roll1);
Serial.print(",");
Serial.print(yaw1);
Serial.print(",");
Serial.print(roll2);
Serial.print(",");
Serial.print(yaw2);
Serial.print(",");
Serial.print(roll3);
Serial.print(",");
Serial.print(yaw3);
Serial.print(",");
Serial.print(pitch3);
Serial.print(",");
Serial.println(envio);
```

Envío de datos

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo para la Transformación de Ángulos RPY a Posiciones XYZ



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Programación para Inicializar los Toolboxes

```
%Inicializar el ToolBox
startup_rvc;
init_lib;
```

Programación para Definir Posiciones Cero del Brazo Robótico

```
%ÁREA DE TRABAJO1
pos_cero_maquina.x1 = 300;
pos_cero_maquina.y1 = 0;
pos_cero_maquina.z1 = 310;
pos_cero_maquina.a1 = 180;
pos_cero_maquina.b1 = 0;
pos_cero_maquina.c1 = 180;
%ÁREA DE TRABAJO2
pos_cero_maquina.x2 = 0;
pos_cero_maquina.y2 = -250;
pos_cero_maquina.z2 = 320;
pos_cero_maquina.a2 = 180;
pos_cero_maquina.b2 = 0;
pos_cero_maquina.c2 = 90;
```

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Recibir Datos de la Tarjeta de Control

```
s = serial(com_serial,'BaudRate',baudios_puerto,'DataBits',8);
%Abrir el puerto serial
timeout = 1;
set(s,'Timeout',timeout);
fopen(s);
```

Configuración y apertura de la comunicación serial

```
valor=fscanf(s,'%d,%d,%d,%d,%d,%d,%d,%d');
```

Recepción de datos

```
%Cambio de variable de los datos obtenidos
%en la comunicación serial
angulo1_roll = valor(1)+calibrar.cal_roll1;
angulo1_yaw = valor(2)+calibrar.cal_yaw1;
angulo2_roll = valor(3)+calibrar.cal_roll2;
angulo2_yaw = valor(4)+calibrar.cal_yaw2;
angulo3_roll  = valor(5)+calibrar.cal_roll3;
angulo3_yaw = valor(6)+calibrar.cal_yaw3;
angulo3_pitch = valor(7)+calibrar.cal_pitch3;
simu_real = valor(8);
```

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Programación para el Cálculo de Coordenadas Rectangulares XYZ

```
%link (revolute , d ,a, alpha)

L(1) = Link ([0 0 0 -pi/2])
L(2) = Link ([0 0 0.34 -pi/2 ])
L(3) = Link ([0 0 0 pi/2 ])
L(4) = Link ([0 0 0.33 pi/2 ])
%0.33 = 0.23(longitud del antebrazo) + 0.1(longitud de la
%muñeca hacia el T-skin

robo = SerialLink (L , 'name' , 'Brazo_Jeff')

qf1= [yaw1 roll_1 yaw2 roll_2];

T = robo.fkine(qf1);

coor_trabajo_X = T.t(1);
coor_trabajo_Y = T.t(2);
coor_trabajo_Z = T.t(3);
%Conversion amm
diferencia_X = pos_x*1000-coor_trabajo_X*1000;
diferencia_Y = pos_y*1000-coor_trabajo_Y*1000;
diferencia_Z = pos_z*1000-coor_trabajo_Z*1000;
posicionX_real =diferencia_X+pos_cero_maquina.x;
posicionY_real =diferencia_Y+pos_cero_maquina.y;
posicionZ_real =diferencia_Z+pos_cero_maquina.z;
```

Parámetros DH del brazo humano

Carga de los parámetros DH

Vector qf1 conformado por 2 sensores (hombro y codo)

Cálculo de coordenadas XYZ

Coordenadas XYZ finales

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

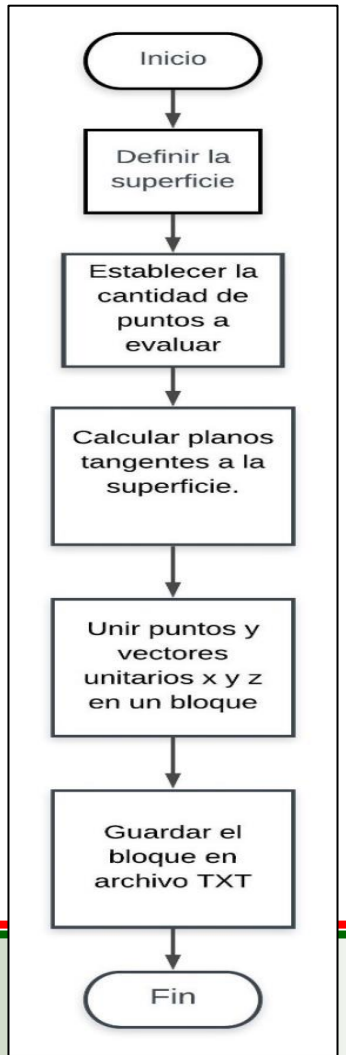
Programación para el Cálculo Coordenadas Angulares ABC

```
roll_pitch_yaw =
tr2rpy(roty(deg2rad(angulo3_roll))*rotx(deg2rad(pos_cero_maquina.a+
angulo3_pitch))*rotz(deg2rad(pos_cero_maquina.a+angulo3_pitch)),'deg');
disp(roll_pitch_yaw);
posicionA_real = roll_pitch_yaw(1);
posicionB_real = roll_pitch_yaw(2)+pos_cero_maquina.b;
posicionC_real = roll_pitch_yaw(3);
```

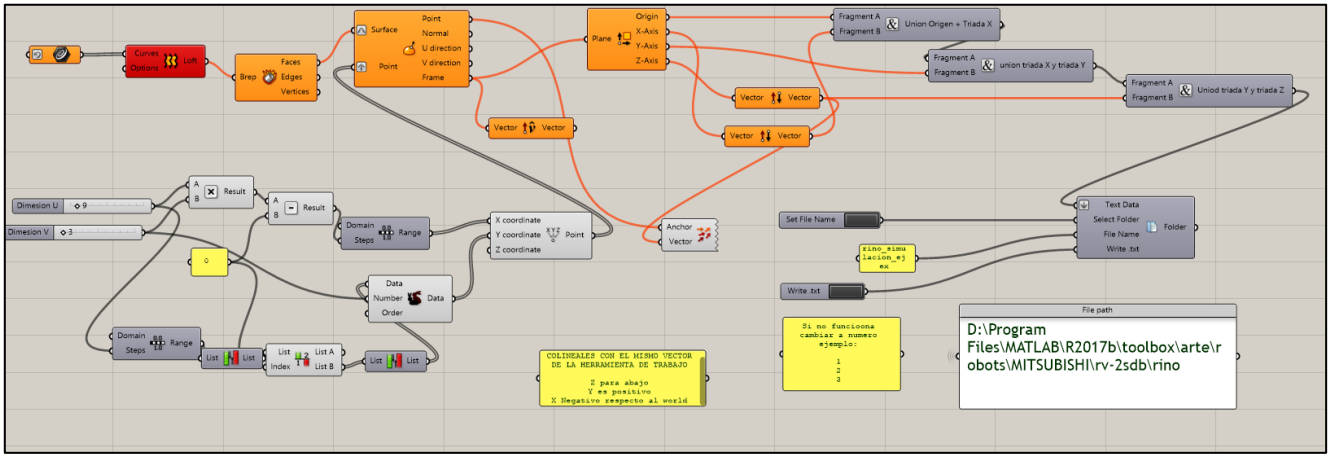
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

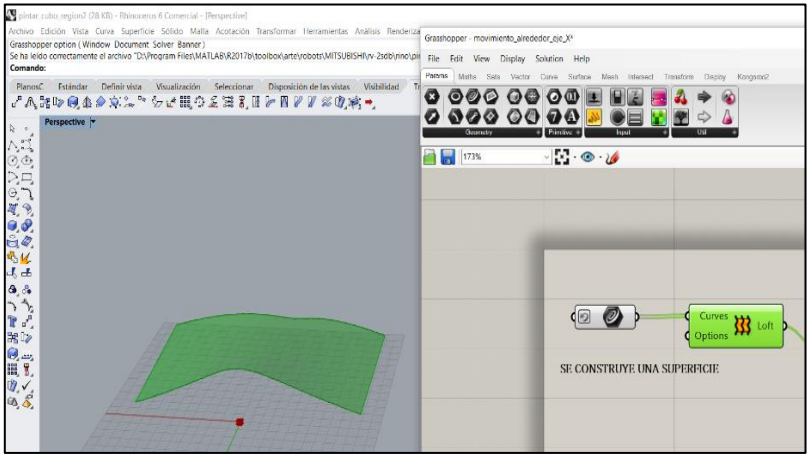
Programación Offline con Rhinoceros

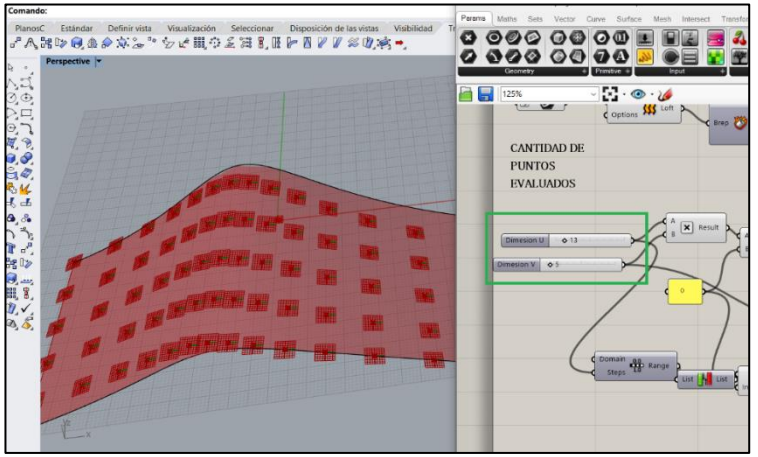
Algoritmo para Obtener Puntos y Vectores Unitarios de una Superficie Compleja.



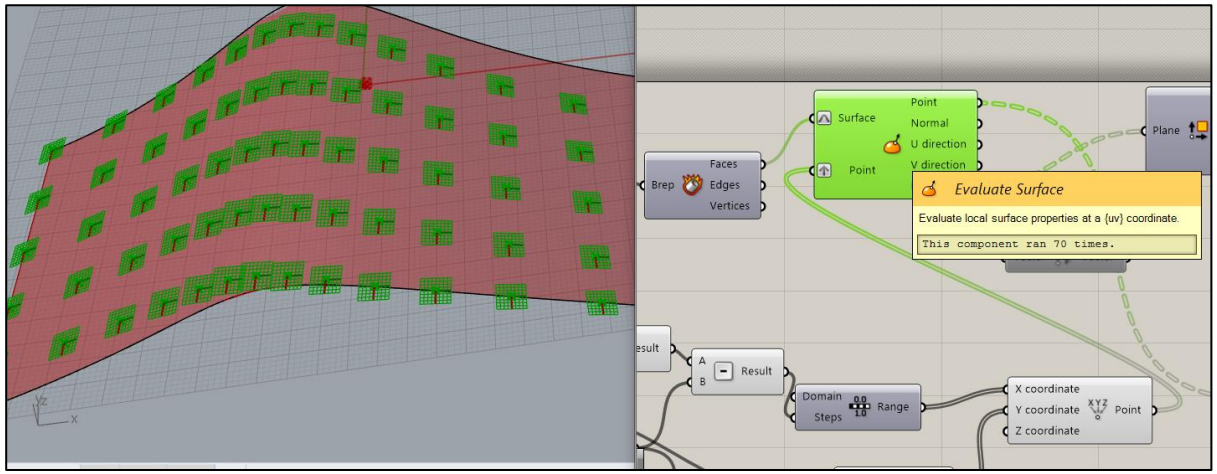
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

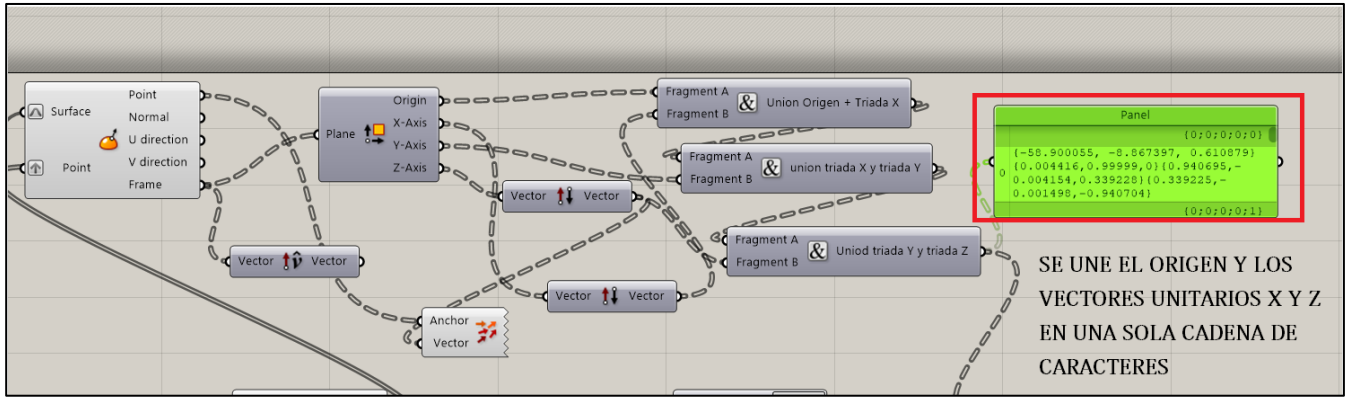




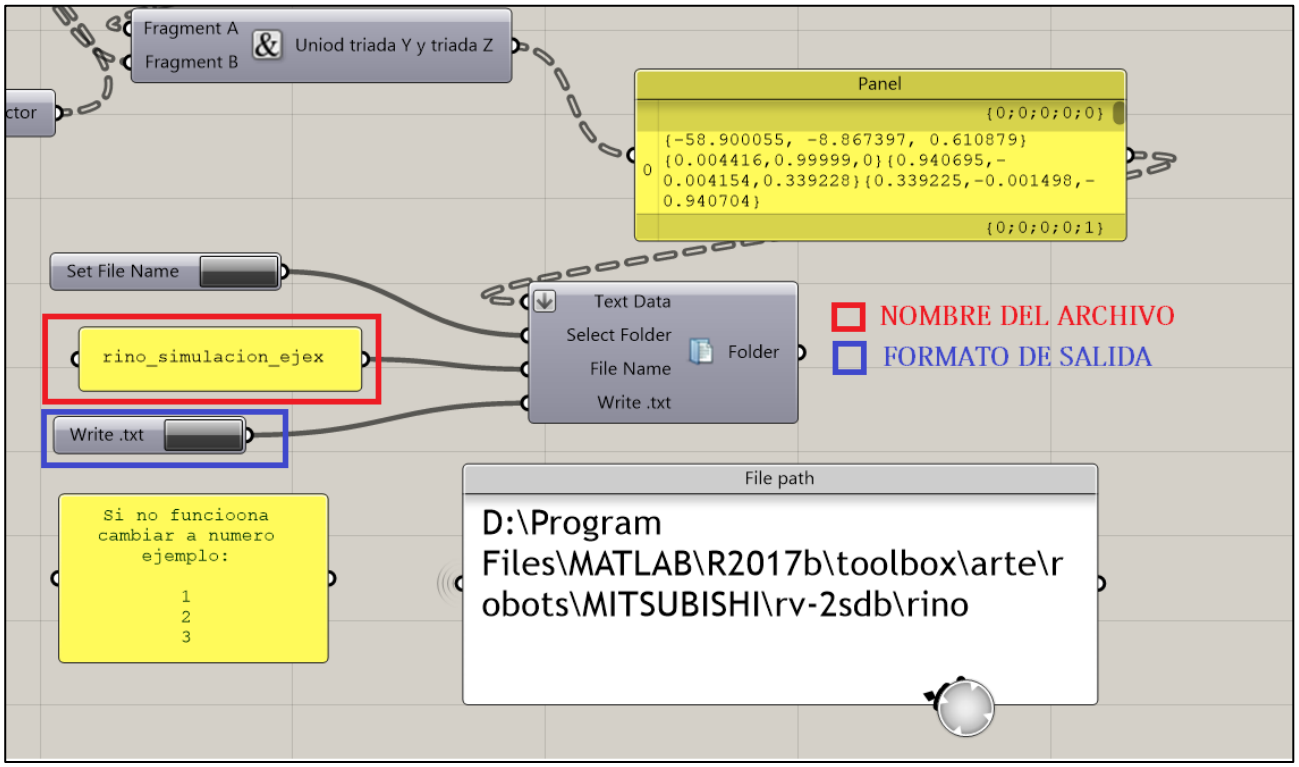


INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL



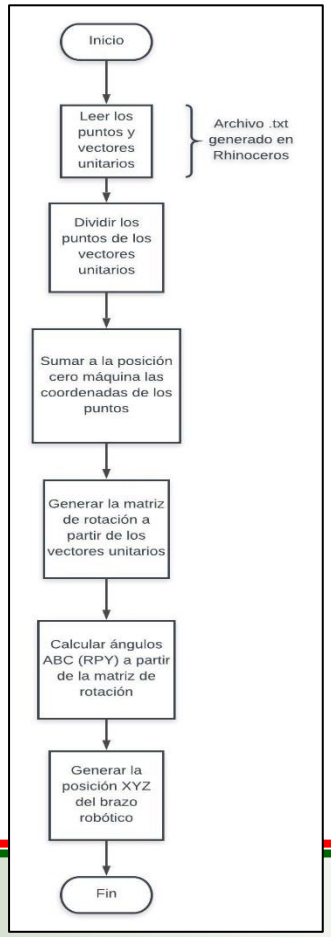


INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

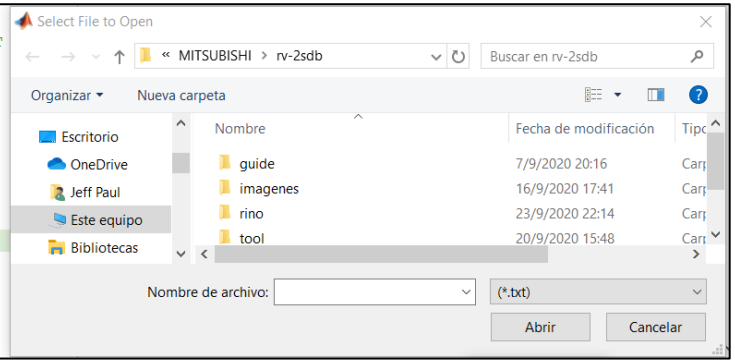
Algoritmo para la Transformación de Puntos y Vectores Unitarios Obtenidos en RHINOCEROS a Posiciones XYZ



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Programación para la Transformación de Puntos y Vectores Obtenidos en Rhinoceros en Posiciones XYZ

```
[file,path] = uigetfile('*.txt'); %PERMITE LA CARGA DE ARCHIVOS TXT
if isequal(file,0)
   disp('Cancelado');
else
   disp(['Archivo Seleccionado ', fullfile(path,file)]);
end
cd (path);
filename = file;
delimiterIn = ',';
datos = importdata(filename,delimiterIn) %DATOS IMPORTANDOS
datos(1);
puntos = 1;
tamano = length(datos);

%ENvio al brazo robotico
```




```
origen(puntos,:) = origen(puntos,:) +[pos_cero_maquina.x pos_cero_maquina.y pos_cero_maquina.z];
```

```
matriz_rot = [vector_normalx(puntos,:);vector_normaly(puntos,:);vector_normalz(puntos,:)];
abc= tr2rpy(matriz_rot,'deg');
```

```
origen_abc(puntos,:) = [origen(puntos,:) abc];
posiciones_xyz = origen_abc(puntos ,:)
```

UNIÓN DE COORDENDAS XYZ CON COORDENADAS ABC

```
[junturas,pos xyz] = inversa cine filtrado simulacion(posiciones xyz(1),
posiciones xyz(2),posiciones xyz(3),posiciones xyz(4),
posiciones_xyz(5),posiciones_xyz(6),puntos,herramienta);
```

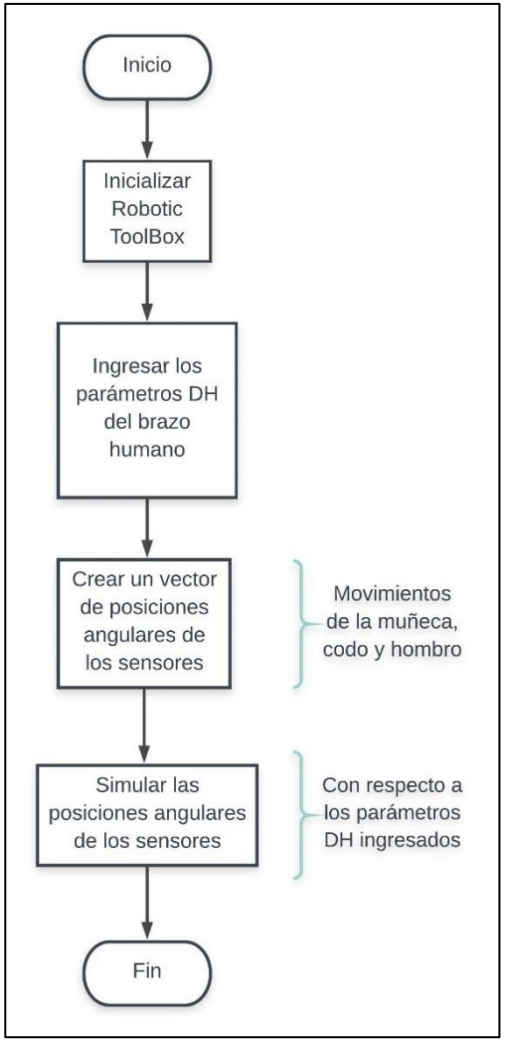
FUNCIÓN QUE PERMITE EL FILTRADO DE LAS POSICIONES XYZ HACIA EL CONTROLADOR

```
posiciones_xyz_envio(puntos,:) = [round(puntos) pos_xyz']
```

POSICIONES QUE PUEDEN SER ENVIADOR AL CONTROLADOR

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Robotics Toolbox como Simulador del Movimiento con Respecto al Brazo Humano



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Programación para Simular el Movimiento del Brazo Humano Utilizando Robotics Toolbox.

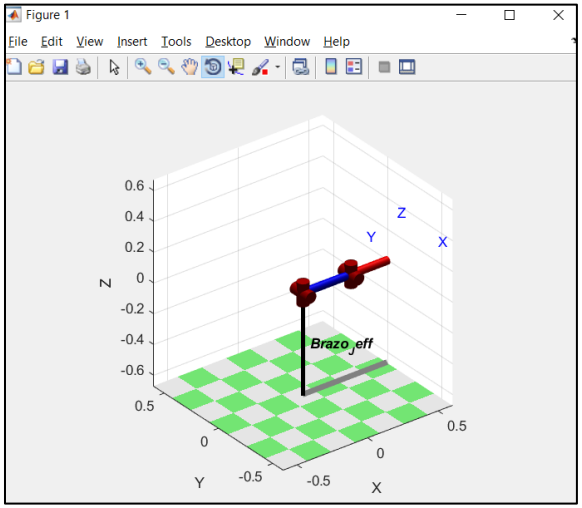
```
startup_rvc;
```
INICIALIZAR EL TOOLBOXS ROBOTICS

```
%link (revolute , d ,a, alpha)

L(1) = Link ([0 0 0 -pi/2])
L(2) = Link ([0 0 0.34 -pi/2 ]
L(3) = Link ([0 0 0 pi/2 ])
L(4) = Link ([0 0 0.33 pi/2 ])
%0.33 = 0.23(longitud del antebrazo) + 0.1(longitud de la
%muñeca hacia el T-skin
%Iniciacion del robot

robo = SerialLink (L , 'name' , 'Brazo_Jeff')
%%Cinematica directag
qf0 = [ 0 0 0 0];
robo.plot(qf0);

yaw1 = deg2rad(angulo1_yaw);
roll_1 =deg2rad(angulo1_roll);
yaw2 =deg2rad(angulo2_yaw);
roll_2 =deg2rad(angulo2_roll);

qf1= [yaw1 roll_1 yaw2 roll_2];
robo.animate(qf1)
```

INGRESAR LOS PARÁMETRO DH PROPIOS DEL BRAZO

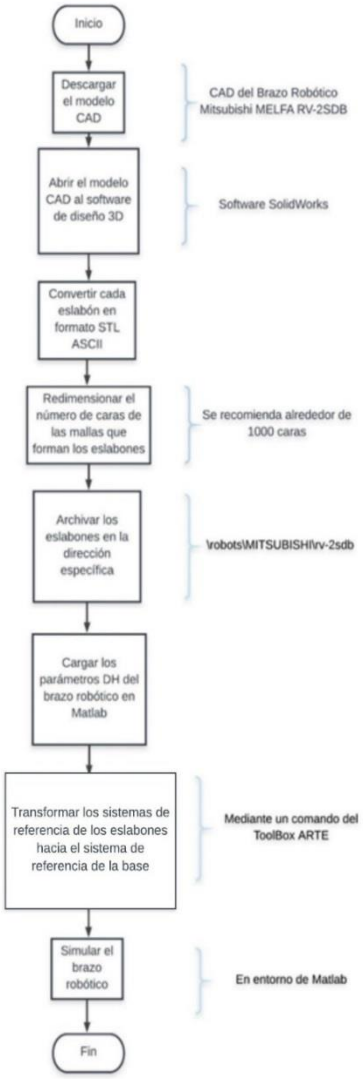
CARGAR LAS CONFIGURACIONES Y SIMULAR EN LA POSICIÓN ORIGINAL

CONVERSIÓN DE GRADOS A RADIANES PARA SU POSTERIOS SIMULACIÓN CON EL VECTOR *qf1*



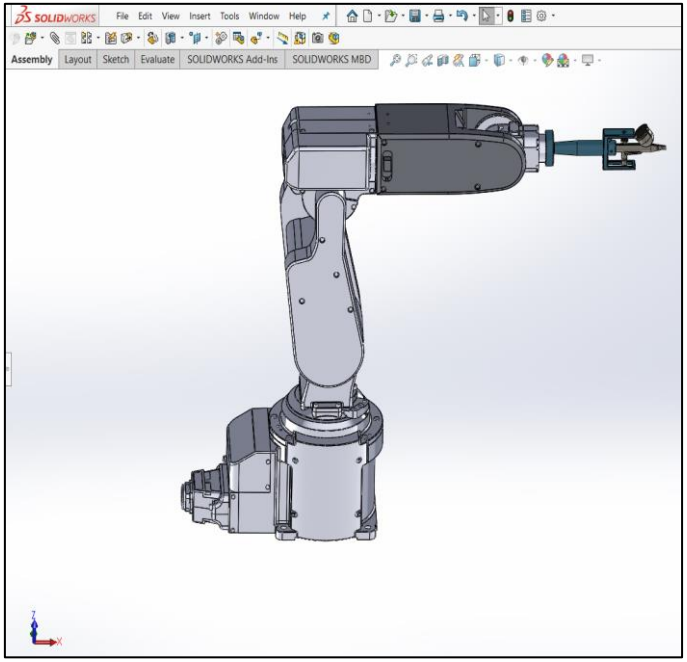
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo para Utilizar ARTE como Simulador del Brazo Robótico.

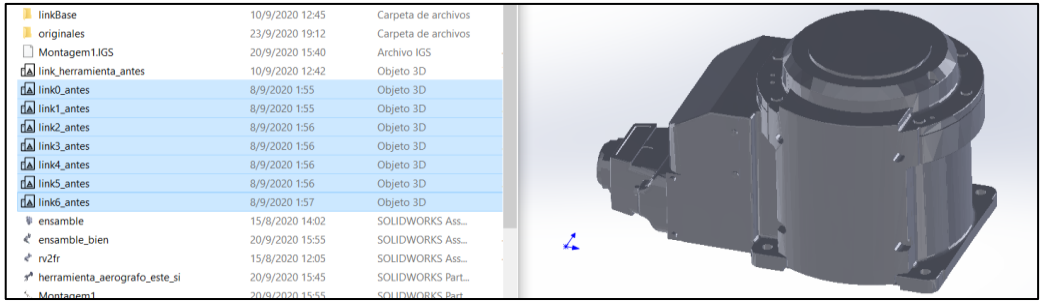


INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Abrir el Modelo CAD

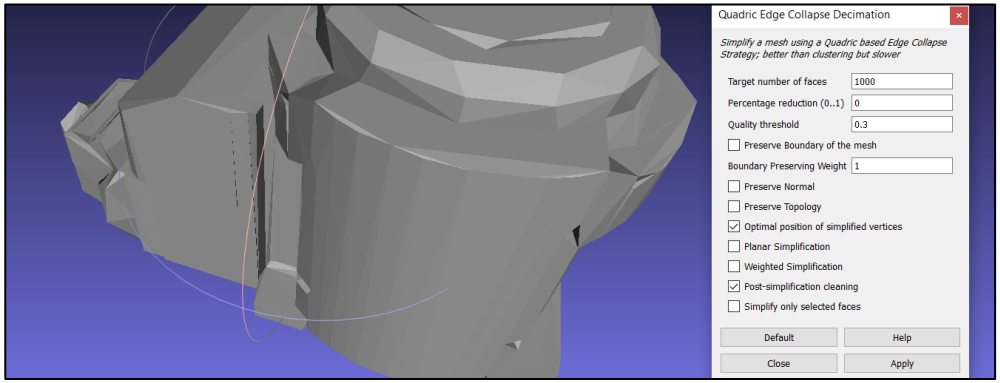


Convertir los Eslabones del Brazo Robótico a Formato STL

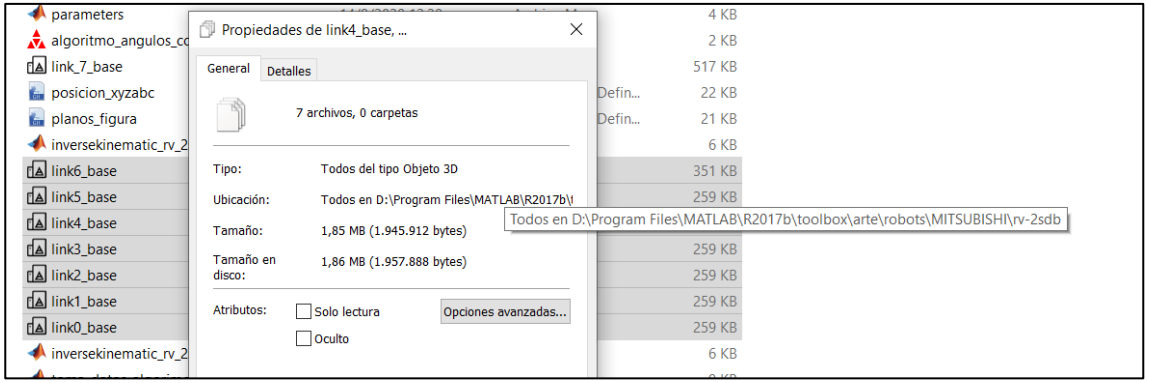


INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Redimensionar el Número de Caras de las Mallas que Forman los Eslabones



Archivar los Eslabones en la Dirección Específica



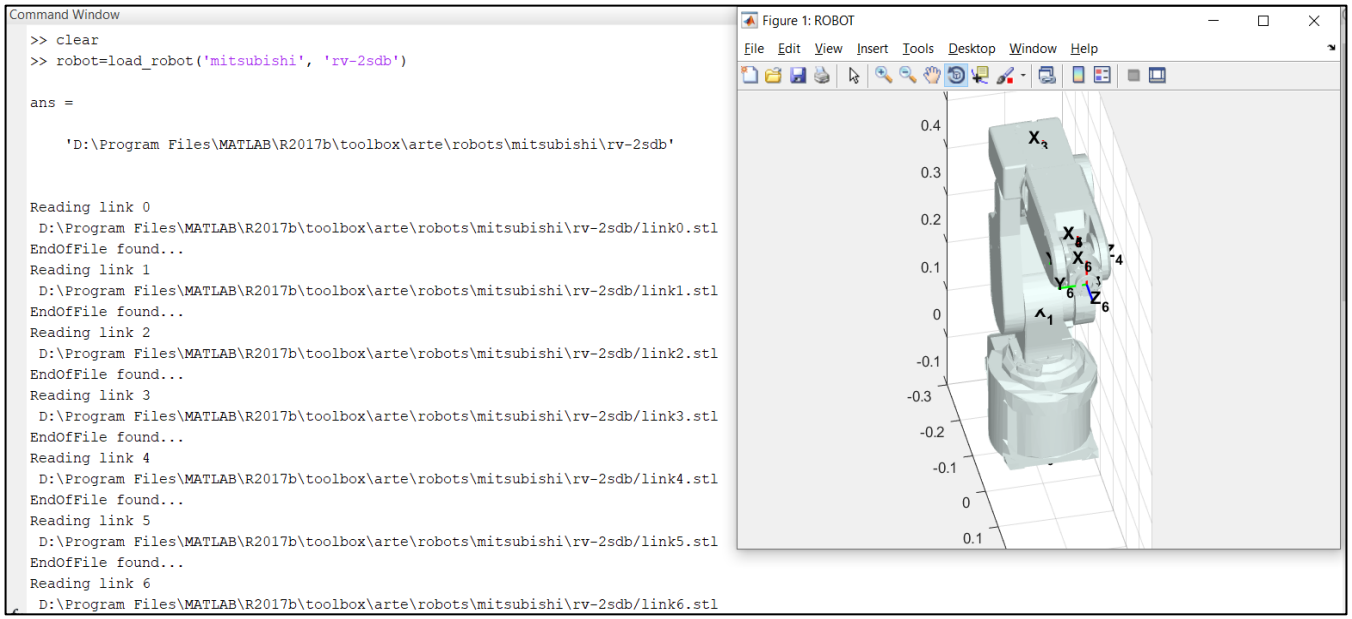
Cargar los Parámetros DH del Brazo Robótico en MATLAB

```
%kinematic data
%robot.DH.theta= '[q(1) q(2)-pi/2 q(3)+pi/2 q(4) q(5) q(6)]';
robot.DH.theta= '[q(1) q(2)-(pi/2) q(3) q(4) q(5) q(6)]';
robot.DH.d='[0.295 0 0 0.270 0 0.07]';
robot.DH.a='[0 0.23 0.05 0 0 0]';
robot.DH.alpha= '[-pi/2 0 -pi/2 pi/2 -pi/2 0]';


% robot.DH.theta= '[q(1) q(2)-pi/2 q(3) q(4) q(5) q(6)]';
% robot.DH.d='[0.400 0 0 0.285 0 0.085 ]';
% robot.DH.a='[0 0.340 0.05 0 0 0]';
% robot.DH.alpha= '[-pi/2 0 -pi/2 pi/2 -pi/2 0]';
%number of degrees of freedom
robot.DOF = 6;

%rotational: R, translational: T
robot.kind=['R' 'R' 'R' 'R' 'R' 'R'];

%Jacobian matrix
robot.J=[];
```

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

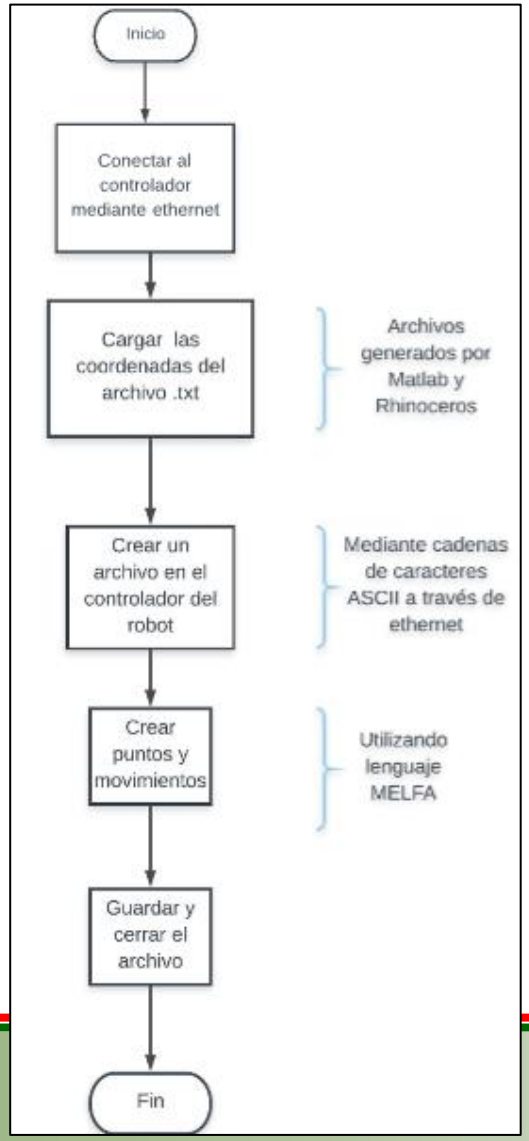
Transformar los sistemas de referencia de los eslabones al sistema de referencia de la base

```
transform_to_own('MITSUBISHI','rv-2sdb\tool',1000)
```



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo para el Envío de Posiciones hacia el Controlador Mitsubishi CR1DA-711



Algoritmo para el Envío de Posiciones hacia el Controlador Mitsubishi CR1DA-711

Función	Comando	Ejemplo	Retorno
Abre la comunicación hacia el controlador	OPEN	1;1;OPEN=USERTOOL	QoK3F;3F;7,0;3,5,A,1E,32,46,64;MPRM;RV-4A;CRn-5xx;MELFA;03-19;Ver.J4;ENG; COPYRIGHT(C)19 2003 MITSUBISHI ELECTRIC CORPORATION ALL RIGHTS RESERVED;1;1;8;
Carga un archivo para editarlo	LOAD=<Nombre del programa>	1;1;LOAD=100	QoK
Crea una variable pueden ser enteros, posiciones xyz o por juntas.	VAL=<Nombre de la variable>=<Valor >	1;9;VAL=M1=3	Qok
Permite la ejecución de comandos MELFA-BASIC IV	EXEC<Instrucción en el lenguaje MELFA-BASIC IV o >	1;1;EDATA10 MOV P1	Qok
Enciende los servos	SRV<ON/OFF>	1;1;SRVON	Qok
Muestra la posición del efector final en distintos tipos.	<Tipo>POS<información> **Tipo:** J: Juntas P: XYZ X:3-axis R: Cilíndricas **Información:** 1-8: Un solo eje F: Todos los ejes	1;1;PPOSF	QoKX;290.62;Y;-0.09;Z;11.26;A 179.94;B;-0.26;C;179.93;L1;0.00;;7,0;100;0.0(000000

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo para el Envío de Posiciones hacia el Controlador Mitsubishi CR1DA-711

```
A = importdata(filename,delimiterIn,headerlinesIn)
datos = A.data;
```

Carga los datos de un txt

```
fwrite(t,'1;1;SAVE'); %Se guarda cualquier arvhico cargado
pause(0.25)
fwrite(t,'1;1;OPEN=USERTOOL'); %abre la comunicacion
pause(0.25);
fwrite(t,'1;1;CNTLON'); %inicia el movimiento en el brazo
pause(0.25);
fwrite(t,'1;1;FDEL2'); %borra el archivo con nombre 2
pause(0.25)
fwrite(t,'1;1;SRVON');%enciende los servos en el brazo
pause(2);
fwrite(t,'1;1;LOAD=2'); %abre el arcihvo en el que se almacena las variables
pause(0.25)
fwrite(t,'1;1;OVRD=25');%velocidad porcentaje
pause(0.25);
fwrite(t,'1;1;RSTALRM');%resetea las alarmas
```

COMUNICACÍON CON STRINGS MEDIANTE ETHERNET

```
cm=['1;9;VAL=P' num2str(puntos,'%d') ' = (' num2str(posiciones_xyz(1),'%10.2f') ', '
    num2str(posiciones_xyz(2),'%10.3f') ', '  num2str(posiciones_xyz(3),'%10.3f')  ', '
    num2str(posiciones_xyz(4),'%10.3f') ', '  num2str(posiciones_xyz(5),'%10.3f') ', '
    num2str(posiciones_xyz(6),'%10.3f') ')(7,0)']
fwrite(t,cm);
cm2=['1;9;EXEC MVS P' num2str(puntos,'%d') '']
fwrite(t,cm2);
pause(1.5);
cm3 = ['1;1;EDATA ' num2str(puntos,'%d') ' MVS P' num2str(puntos,'%d') '']
fwrite(t,cm3);
```

CREAR PUNTOS Y MOVIMIENTOS CON LA ESTRUCTURA DEL LENGUAJE MELFA IV

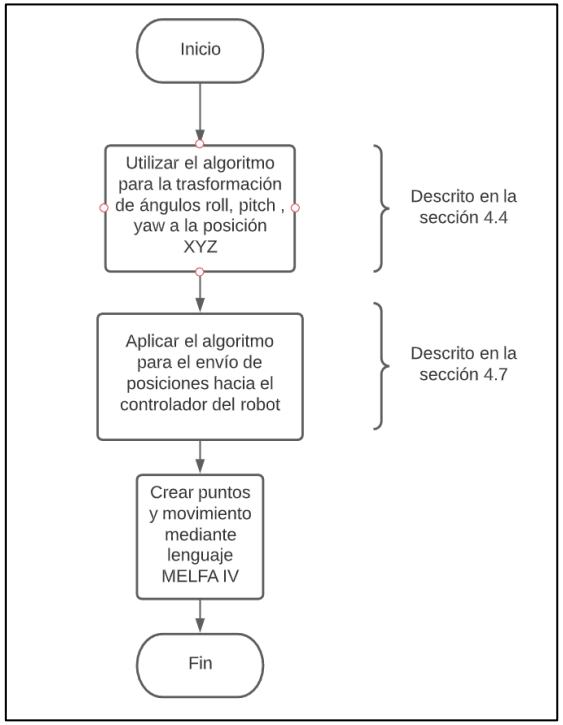
```
fwrite(t,'1;1;SAVE'); %guarda el archivo modificado
pause(0.25)
fwrite(t,'1;1;SRVOFF');%apaga los servos
pause(0.25)
fwrite(t,'1;1;CNTLOFF'); %apaga el movimiento en el brazo
```

INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo que Utiliza el HMI para el Control y Programación del Brazo Robótico

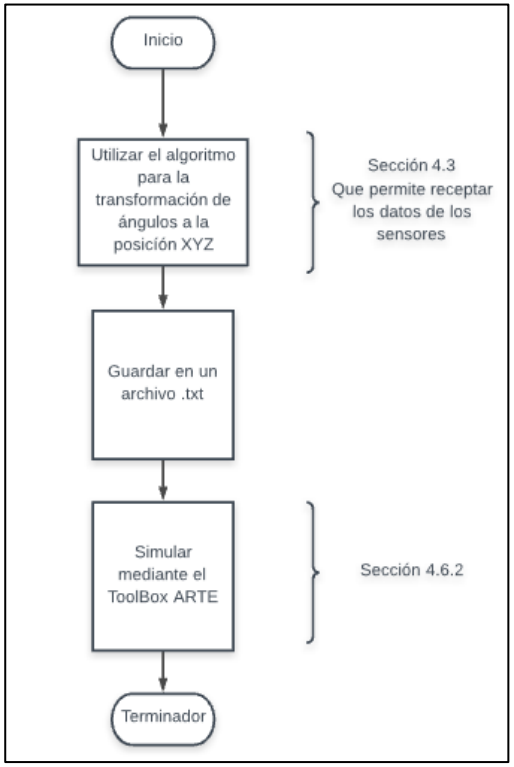
Mitsubishi

Algoritmo para el Tiempo Real



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo para la Programación Mediante Movimientos Biomecánicos




```
A = importdata(filename,delimiterIn,headerlinesIn);
datos = A.data
```

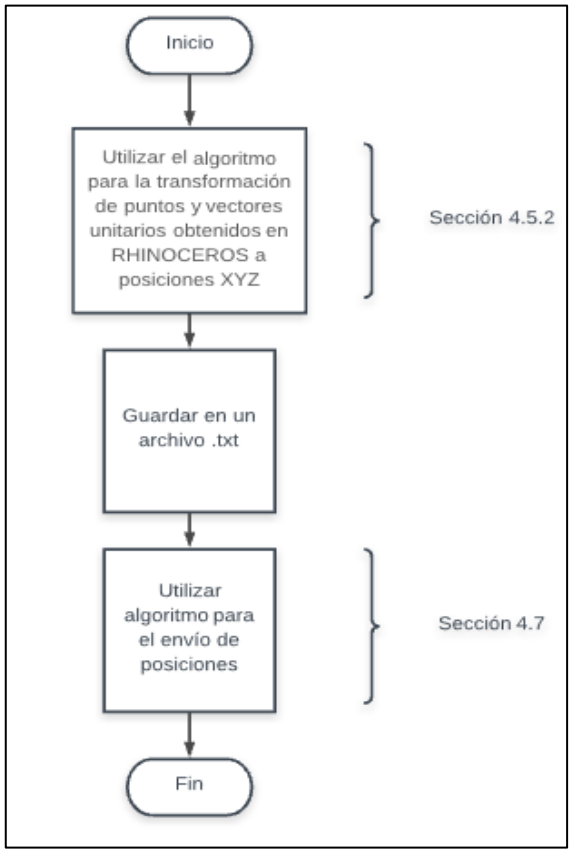
DATOS DEL ARCHIVO TXT

```
junturas = datos(:,puntos);
drawrobot3d(robot_inver,deg2rad(junturas));
```

SIMULACIÓN UTILIZANDO ARTE

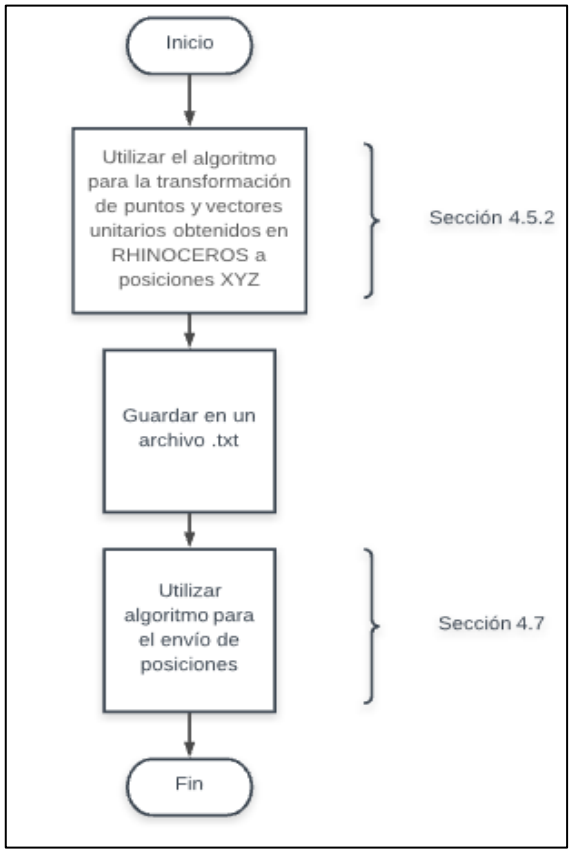
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo para la Programación Offline con Rhinoceros



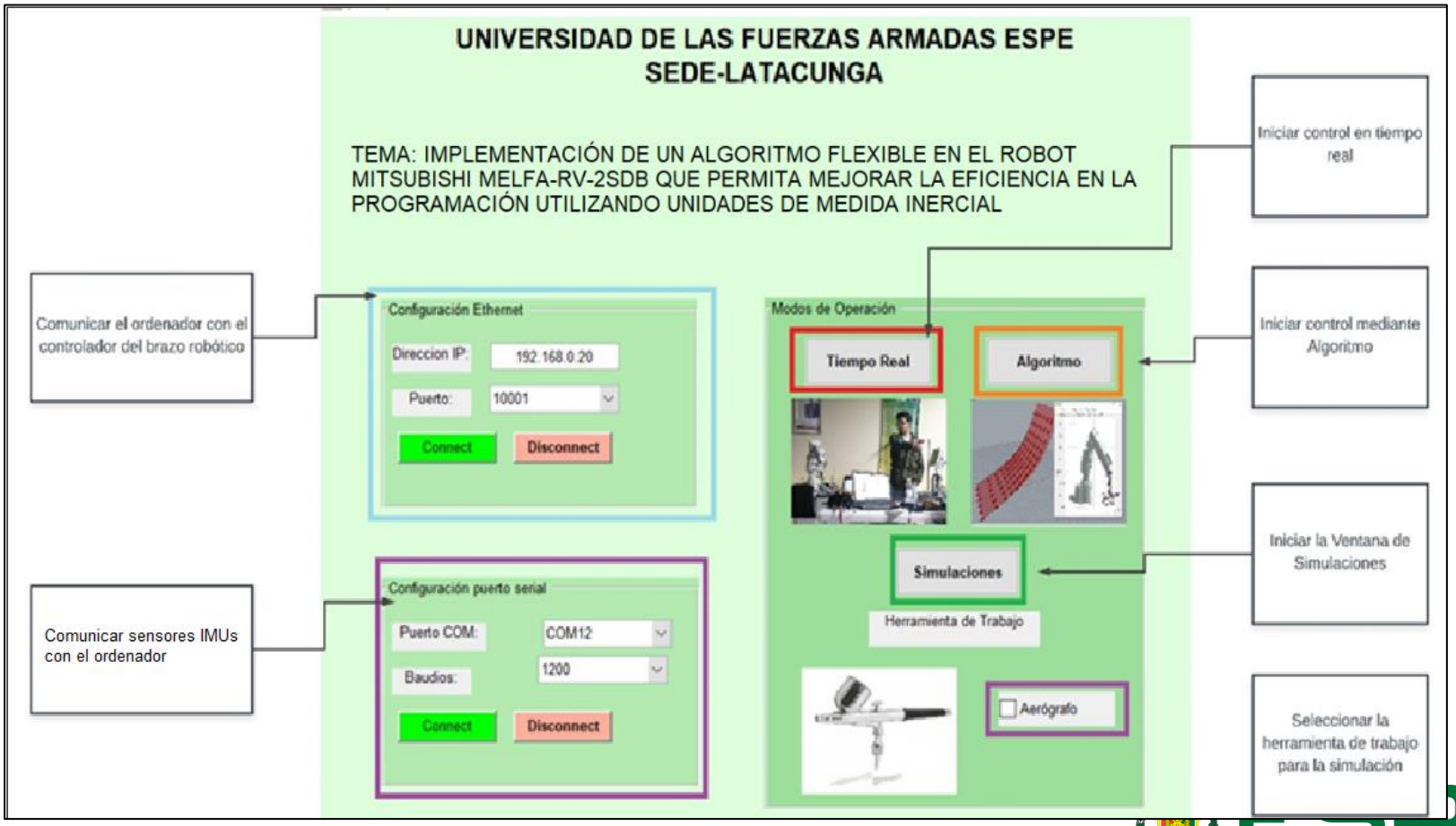
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Algoritmo para la Programación Offline con Rhinoceros



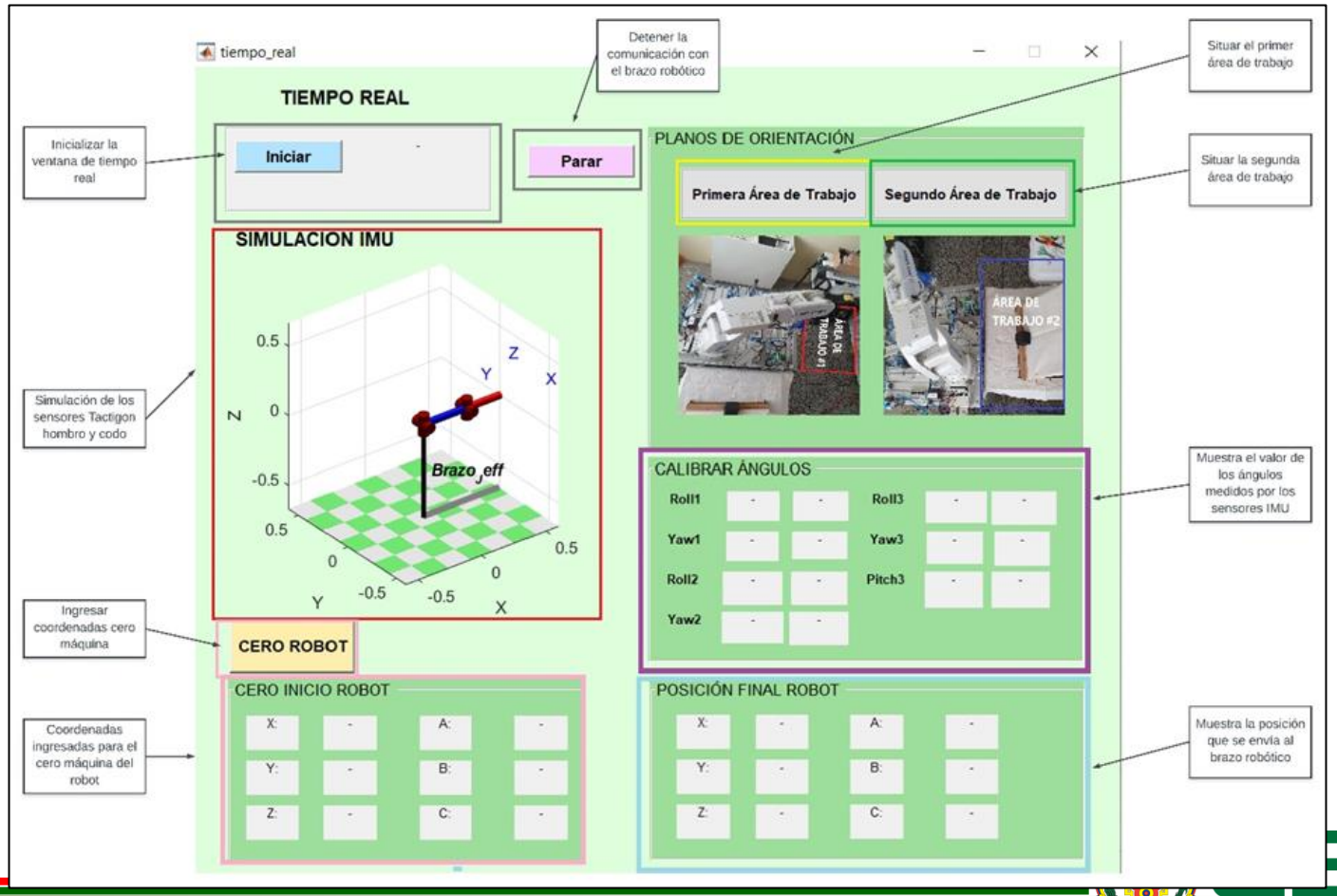
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Implementación de la interfaz para la Comunicación del Algoritmo Flexible



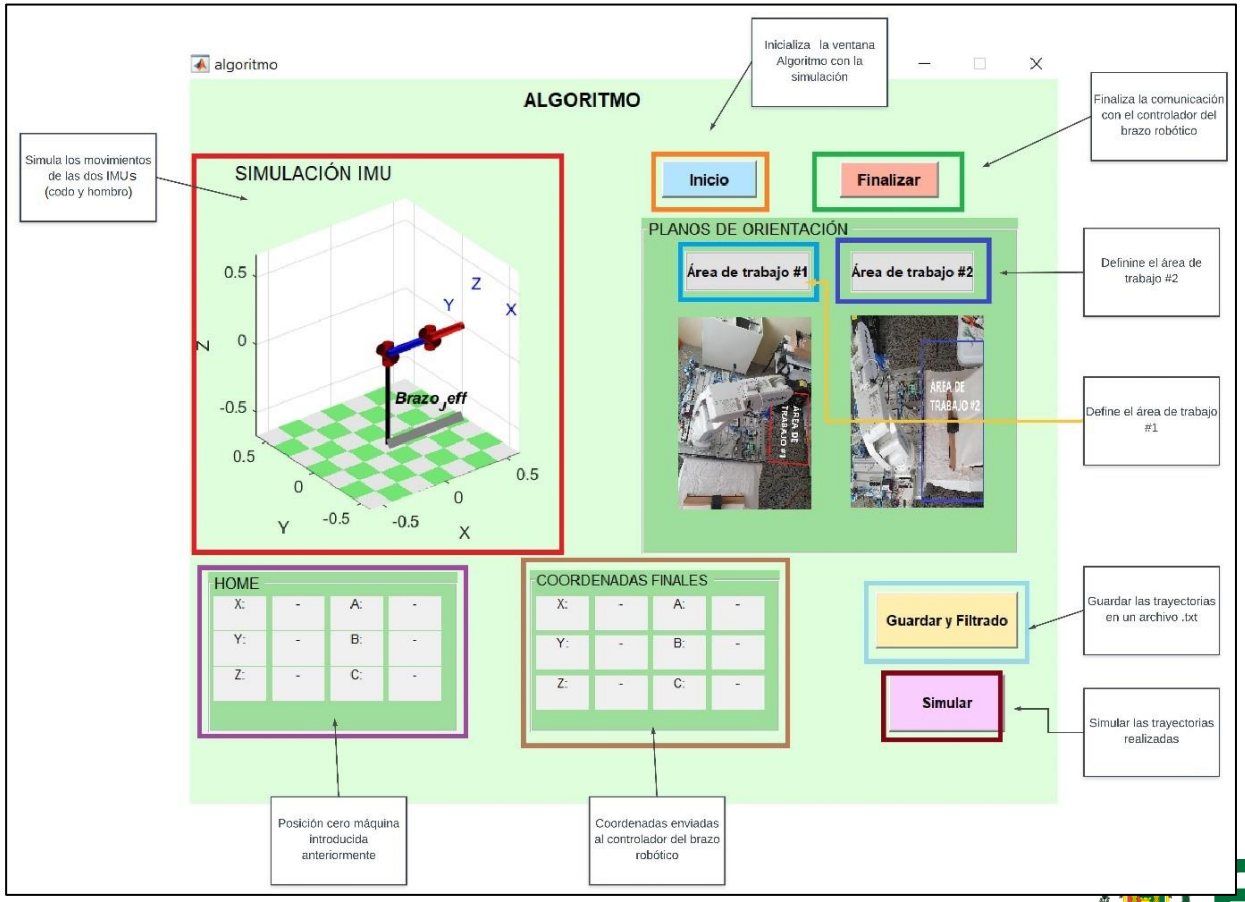
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Implementación de la interfaz para la Comunicación del Algoritmo Flexible



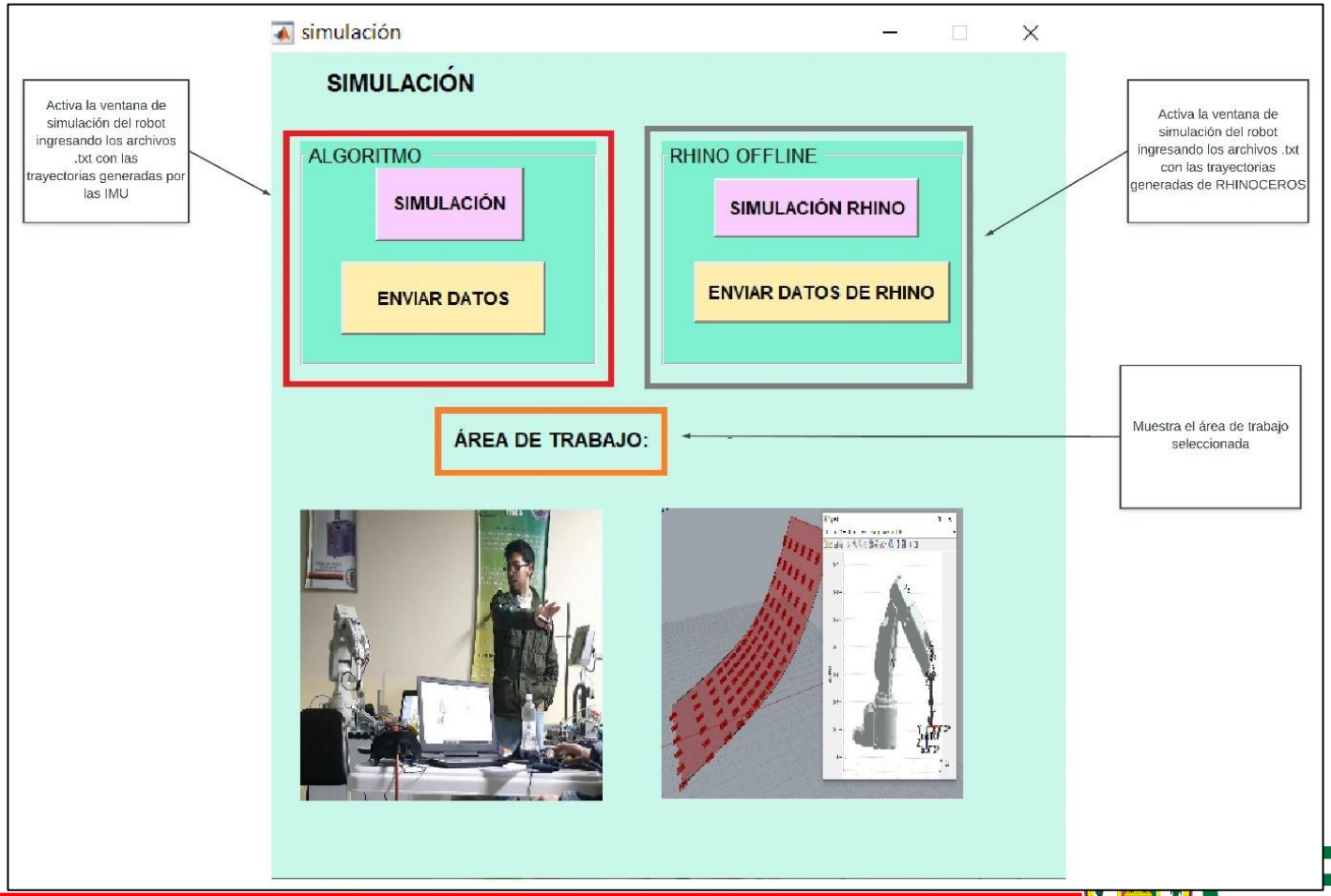
INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Implementación de la interfaz para la Comunicación del Algoritmo Flexible



INTEGRACIÓN DE SISTEMAS E IMPLEMENTACIÓN DEL ALGORITMO DE CONTROL

Implementación de la interfaz para la Comunicación del Algoritmo Flexible



RESULTADOS OBTENIDOS Y ANÁLISIS

Precisión y exactitud

Pruebas de Biomecánica

Movimiento	Posición
Flexión	
Extensión	
Abducción	
Aducción	
Rotación medial	
Rotación lateral	

Pruebas de Biomecánica

Movimiento	Posición
Flexión	
Extensión	
Pronación	
Superación	 

Pruebas de Biomecánica

Tabla 23

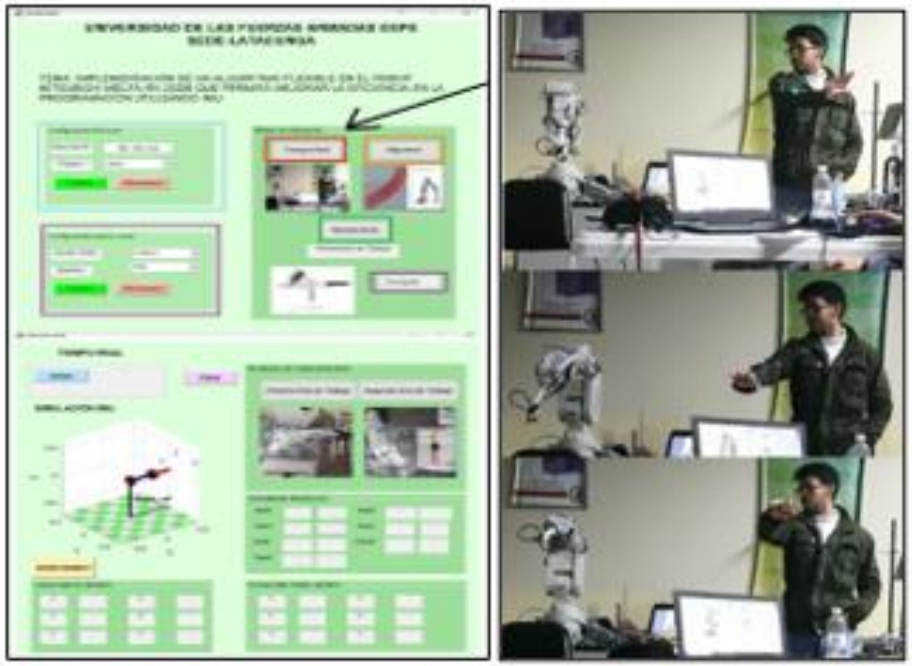
Biomecánica de la muñeca

Movimiento	Posición
Extensión	
Flexión	

Pruebas de Modo Tiempo Real

Figura 87

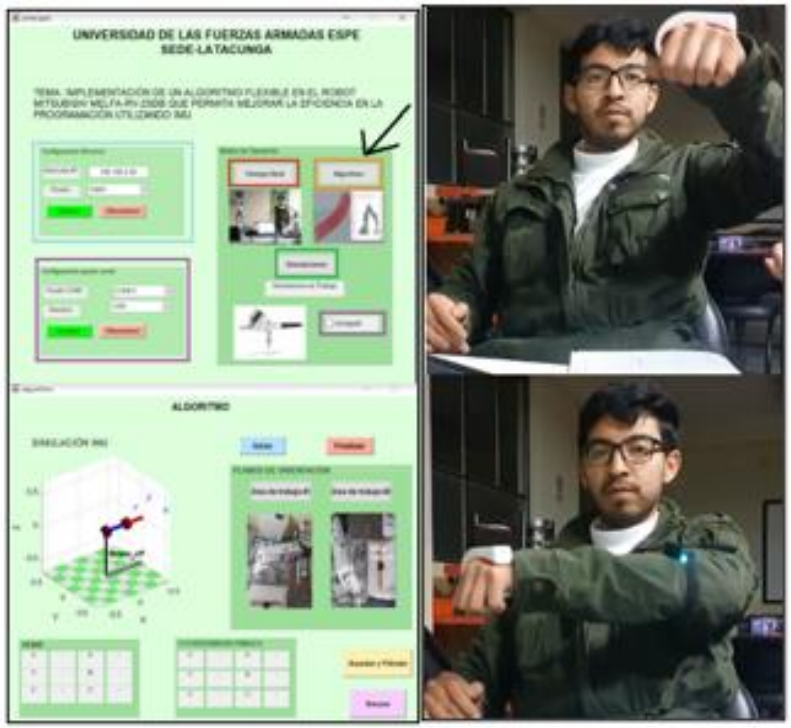
Modo Tiempo Real



Pruebas de Modo Algoritmo Flexible

Figura 88

Modo Algoritmo Flexible



Validación de la Hipótesis

$$Eficiencia = \frac{Tiempo\ Normal\ (Teach\ Pendant)}{\frac{Tiempo\ de\ trayectoria\ generada\ (Algoritmo\ Flexible)}{Calificación}} x100\%$$

$$Umbral_{eficiencia} = 100\%$$

Se tiene que: $Si\ Eficiencia > Umbral_{eficiencia}\ y\ Calificación > 0.6$

Solo si cumple ambas condiciones entonces el Resultado es Positivo ✓ , caso contrario

Resultado Negativo ✗.

Validación de la Hipótesis

Resultado/trayectoria	Simples		Circulares		Total
✓	Línea	9	Función "Seno"	8	13
	Zigzag	8	Círculo	6	11
	Rombo	8	Elipse	6	11
	Número "7"	10	Número "3"	8	14
	Total	**35**	Signo de Interrogación	6	3
			Total	**34**	**69**
✗	Línea	1	Función "Seno"	2	2
	Zigzag	2	Círculo	4	4
	Rombo	2	Elipse	4	4
	Número "7"	0	Número "3"	2	1
	Total	**5**	Signo de Interrogación	4	2
			Total	**16**	**21**
Total	**40**		**50**		**90**

Hipótesis Nula (Ho): El algoritmo flexible no mejora la eficiencia en la programación.

Hipótesis alternativa (Hi): El algoritmo flexible mejora la eficiencia en la programación.

Validación de la Hipótesis

Chi-Cuadrado Calculado (x^2_{calc}): $\sum \frac{(f-ft)^2}{ft}$

$f \rightarrow frecuencia\ observada\ (sumatoria\ de\ los\ datos\ en\ la\ tabla)$

$ft \rightarrow frecuencia\ teórica$

(x^2_{calc}): $$\sum \frac{(f-ft)^2}{ft} = \frac{(35-30.67)^2}{30.67} + \frac{(34-38.33)^2}{38.33} + \frac{(5-9.33)^2}{9.33} + \frac{(16-11.67)^2}{11.67}$$

$$= 0.61 + 0.49 + 2 + 1.61$$

$$x^2_{calc} = 4.71$$

Se trabajó con un nivel de significancia del $\propto = 5\%$. Se calcula entonces la proporcionalidad p con la siguiente fórmula:

$$p = 1 - nivel\ de\ significancia = 0.95$$

Cálculo del Grado de libertad (v):

$$v = (n°\ de\ filas - 1) * (n°\ de\ columnas - 1)$$

$$n°\ de\ filas = 2\ (Resultado\ positivo/negativo)$$

$$n°\ de\ columnas = 2\ (Trayectorias\ Simples/Compleja)$$

$$v = (2-1) * (2-1) = 1$$

Teniendo 1 grado de libertad y para un valor de proporcionalidad del 0.95 se obtiene el calor del Chi Cuadrado en su tabla de distribución:

$$x^2_{tabla} = 3.841$$

Como el Chi-Cuadrado calculado es mayor que el Chi-cuadrado obtenido en la tabla, entonces se procede a descartar la Hipótesis nula. Por tanto, se procede a validar la hipótesis alternativa que sostiene que el algoritmo flexible mejora la eficiencia en la programación.

Conclusiones

- Para el diseño del algoritmo flexible se partió de la recopilación de conceptos que intervienen dentro de la robótica industrial como la cinemática directa e inversa que son datos muy importantes junto a los parámetros Denavit-Hartenberg los cuales permiten el cálculo de posiciones del efector final en cualquier momento, además se realizó una búsqueda de los distintos sensores que manejan unidades de medida inercial encontrando que existen varios modelos como mecánicos, ópticos y tipo MEMS, se determinó que el modelo más eficiente para la aplicación es el MEMS debido a su tamaño, posee la capacidad procesar datos, comunicar y actuar sobre su entorno. Por último, se reúne información acerca de los lenguajes de programación aplicados en el brazo robótico encontrando que poseen una categoría enfocada en la forma de programación y se la puede dividir en programación textual y offline.
- Al momento de adquirir y procesar las señales de las unidades de medida inercial se utilizó un microprocesador integrado con bluetooth, batería, giroscopio, acelerómetro y sensor magnético, estos últimos tres datos son utilizados como datos de entrada para un algoritmo propio del microprocesador permitiendo obtención de los movimientos angulares como el roll, pitch y yaw de la superficie en la que se encuentra sujeta en este caso una extremidad del tren superior.
- Para establecer un enlace entre los componentes del sistema, se diseñó una shield bluetooth capaz de conectar todo el sistema electrónico permitiendo enviar los datos de los sensores hacia el ordenador mediante comunicación serial, estos datos son procesados, transformados y enviados al controlador del brazo robótico través de comunicación ethernet. Una de la funciones del algoritmo flexible es el de convertir los datos de los sensores a un lenguaje de programación del brazo robótico además permite cargar posiciones y movimientos al controlador en modo control externo gracias a la gran variedad de comandos como *“1;1;SRVON”* que permite encender los servomotores, *“1;9;EXECMOV P1”* mueve el efector final hacia la posición *P1* y *“1;9;VAL=P2=(x,y,z,a,b,c)(7,0)”* permite almacenar los valores *x,y,z,a,b,c* en la variable posición *P2*.
- Al momento de diseñar el algoritmo flexible se utilizó un lenguaje estructurado debido a la facilidad que se tiene para añadir funcionalidades permitiendo realizar acciones más complejas. El algoritmo flexible está compuesto por pequeños programas que realizan tareas distintas como es el caso del algoritmo que permite transformar los ángulos de los sensores RPY a posiciones XYZ, el algoritmo para enviar las posiciones hacia el controlador o el algoritmo para la programación offline cada uno de estos algoritmos posee un proceso detallado y se encuentran enlazados mediante el HMI implementado en el ordenador.

Conclusiones

- Se implementó tres modos de trabajo. El primero es el modo tiempo real, el cual permite al operador comprender la relación que existe entre los movimientos del brazo humano y las posiciones enviadas al brazo robótico en tiempo real, este modo no se guarda las posiciones dentro del controlador. El segundo es el modo algoritmo flexible que programa al brazo robótico con los movimientos angulares del brazo humano, permite cambiar el nombre del archivo a guardar en el controlador. Por último, el modo offline que utiliza Rhinoceros como software intermediario para calcular las posiciones y movimientos complejos que se requiera realizar.
- Se realizaron pruebas con el algoritmo flexible por medio de trayectorias simples, circulares y complejas. Para validar cada uno de los ensayos que se realizó, se impuso un umbral de eficiencia, con el fin de medir si el resultado posee una alta eficiencia en tiempo de programación y además la trayectoria generada posee similitud con la maniobra que ejecutó el operador. Las trayectorias simples no representaron problema alguno, teniendo 35 resultados positivos de 40 ensayos; las trayectorias circulares tuvieron 34 resultados positivos de 50 ensayos. El algoritmo flexible permite además programar en cuestión de minutos (programación offline) trayectorias complejas utilizando Software como Rhinoceros, lo que resulta imposible de realizar mediante la programación del brazo robótico normal (Teach Pendant), estos ensayos fueron excluidos del análisis para la validación de la hipótesis planteada. Gracias a los resultados obtenidos se logró relacionar las trayectorias generadas mediante el algoritmo flexible y la eficiencia en la programación, mediante la prueba de Chi-Cuadrado ($x^2_{calc} = 4.71 > x^2_{tabla} = 3.84$) en la que se procedió a tomar la hipótesis alternativa, la cual sostiene que el algoritmo flexible mejora la eficiencia en la programación con un nivel de significancia del 5%.
- Durante el movimiento del robot en las aplicaciones se presentaron ciertas singularidades. Las singularidades son aquellos puntos en los que el robot no puede alcanzar las posiciones y rotaciones calculadas por el algoritmo flexible y se presentan debido a que los ángulos $J_1, J_2, J_3, J_4, J_5\ y\ J_6$ (resultado de las soluciones de la cinemática inversa) no se encuentran dentro del rango de operación del brazo robótico. Las posiciones y rotaciones calculadas se encuentran estrechamente relacionadas con los movimientos de los sensores que son acoplados al brazo humano y son calculadas de la siguiente manera: las posiciones x, y, z dependen de los sensores ubicados en el antebrazo y brazo, mientras que las rotaciones A, B, C dependen únicamente del sensor ubicado en la mano. Se identificó que el sensor ubicado en la mano es el más propenso al envío de singularidades ya que se puede acceder a cualquier punto x, y, z, pero no con todas las rotaciones A, B, C, un claro ejemplo sucede cuando el brazo robótico se encuentra con todas sus junturas formando un ángulo de 0 grados, solo puede acceder al punto más alto $x = 0, y = 0\ y\ z = máx$ cuando $A = 0°\ y\ B = 0°$ (el valor del ángulo C es indiferente), si se añadiese algún desplazamiento angular extra en $A\ o\ B$ ya no se encontraría una solución para $x = 0, y = 0\ y\ z = máx$ lo que caería en una singularidad.

Recomendaciones

- Una vez finalizado con el diseño del sistema electrónico del proyecto se encontró que, para mejorar la interacción entre el operador y los sensores, se debería implementar un módulo bluetooth en la shield para obtener una conexión inalámbrica y no depender de la comunicación serial.
- Para evitar la desconexión entre los sensores y la shield es necesario que el operador se encuentra a distancia máxima de 2 metros entre los dispositivos.
- Para que el operador obtenga buenos resultados en la aplicación de pintura se requiere utilizar en primera instancia el modo tiempo real para conocer los distintos desplazamientos del brazo robótico.

Video

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

IMPLEMENTACIÓN DE UN ALGORITMO FLEXIBLE EN EL ROBOT MITSUBISHI MELFA RV-2SDB QUE PERMITA MEJORAR LA EFICIENCIA EN LA PROGRAMACIÓN UTILIZANDO UNIDADES DE MEDIDA INERCIAL

AUTORES:

PICHUCHO CASTELLANO, JEFFERSON PAÚL

SAMPEDRO GÓMEZ, ANDRÉS MARCELO

DIRECTOR:

ING. MENDOZA CHIPANTASI, DARÍO JOSÉ