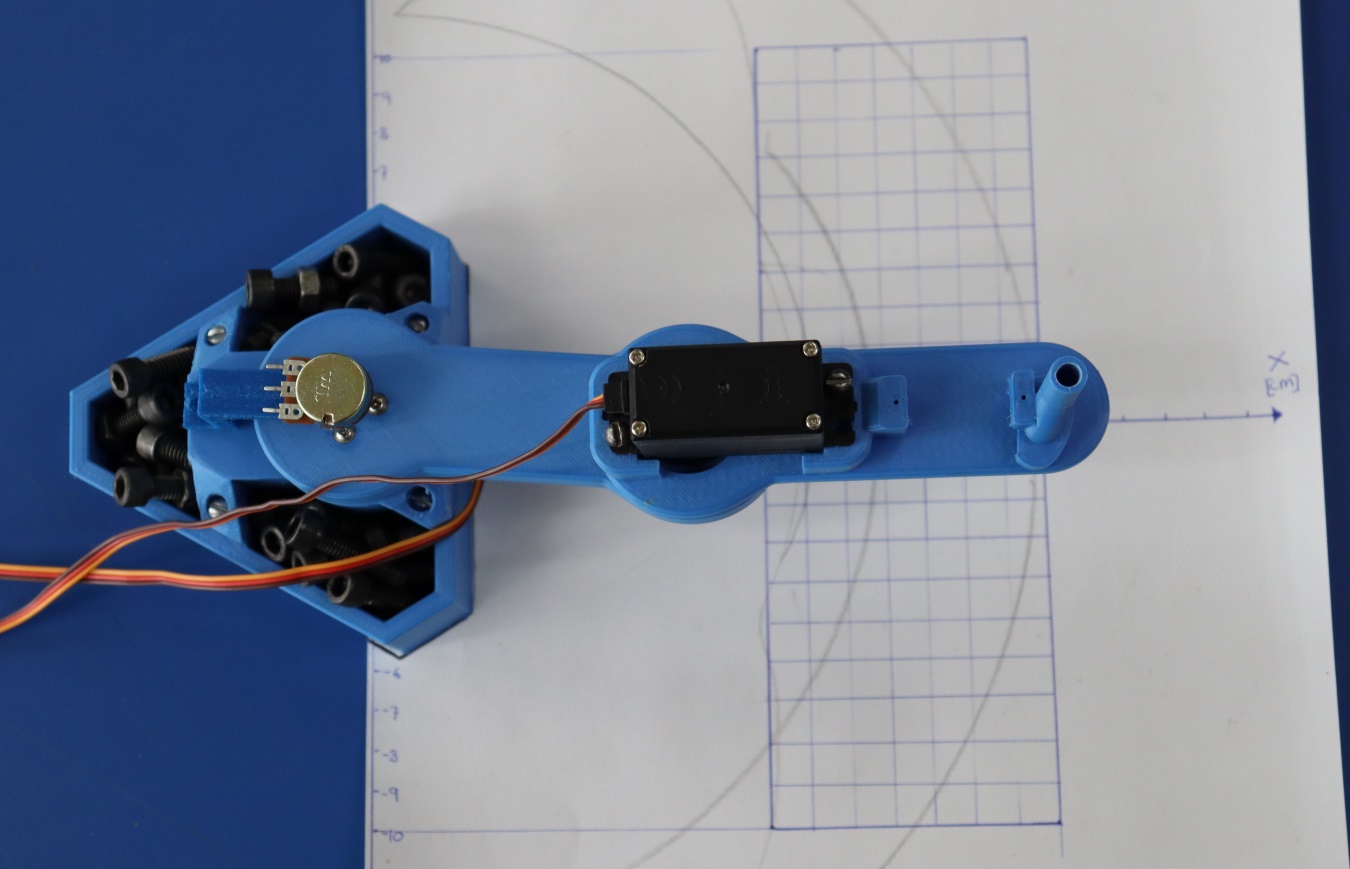
|  |  |
| --- | --- |
|  | |
|  | |
| **Elaborado para:** | Código IoT |
|  |  |
| **Fecha de elaboración:** | 5 de Agosto de 2022 |
| **Vigencia:** | 30 días naturales |
|  |  |
| **Elaborado por:**  **Revisado por:** | Karina Granados Albarrán, Gabriela Ocampo Tovar y  Marco Antonio Ruíz Gasca |
|  | Ing. Hugo Vargas y Mtra. Paloma A. Vilchis León |
| **Documento:** | Reporte final de proyecto |
|  | |

**Control tele operado de brazo robótico para educación a distancia**



**Objetivo general.**

Teleoperar un brazo robótico, reportando los datos de ejecución.

**Objetivos específicos**:

* Diseñar el brazo robótico en un software CAD.
* Imprimir en brazo robótico en una impresora 3D.
* Modelar matemáticamente la cinemática inversa del brazo robótico.
* Realizar la comunicación a distancia con el protocolo de comunicación MQTT.
* Hacer la comunicación a distancia con el protocolo de comunicación MQTT y Node-RED.

**Introducción.**

**Robot Industrial**

De acuerdo a la RIA (Asociación de Industrias Robóticas), un robot industrial es un manipulador multifuncional reprogramable, capaz de mover diferentes piezas, materias, herramientas o dispositivos, de acuerdo a sus trayectorias variables y programadas para realizar dichas tareas.

Y según la Organización Internacional de Estándares (OIE), habla sobre un manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas con el fin de realizar diversas tareas (Salazar Patin, 2015).

**Clasificación del Robot Industrial.**

La Asociación Francesa de Robótica Industrial (AFRI), Califica a los Robots industriales en los siguientes tipos y generaciones:

1. Tipo A: Manipulador de Control Manual o Telemando.
2. Tipo B: Manipulador automático con ciclos preajustados, regulados mediante fines de carrera, controlados por PLC y de accionamientos neumático, eléctrico o hidráulico.
3. Tipo C: Programable de trayectoria continua o punto a punto y carece de conocimiento de su entorno.
4. Tipo D: Robot capaz de adquirir datos de su entorno, readaptando su tarea en función de estos.
5. 1era Generación: Robots Manipuladores, Repite la tarea programada secuencialmente.
6. 2da Generación: Robots De Aprendizaje, Adquiere información limitada de sus entrono y actúa en consecuencia. Capaz de clasificar, detectar esfuerzos, localizar y adaptar sus movimientos en consecuencia a lo que necesite.
7. 3ra Generación: Robots Con Control Sensorizado, Pose una programación que se realiza mediante un lenguaje neutral, el cual le permite una planificación automática de sus tareas, captando su entorno mediante sensores.

**Estructura del Brazo Robótico.**

Los brazos robóticos industriales, tienen la siguiente tipología.

*Articulaciones*

Es la interconexión de dos piezas u objetos que permiten una unión, permitiendo que se mueva relativa a la otra y se conocen cinco tipos de articulaciones cm se puede observar en la Figura 1:

* Articulación Prismática: Formada por dos uniones anidadas, que se desplazan dentro y a lo largo de cada una. El movimiento que presenta es relativo en las uniones, el cual produce un movimiento en línea recta (Extendiéndose o retrayéndose).
* Articulación Rotacional: Las articulaciones de revolución permiten que una unión gire sobre un único eje en el otro, como una puerta y una bisagra.
* Articulación Cilíndrica: Consiste en una articulación de rotación y una articulación de traslación y poseen dos grados de libertad.
* Articulación Planar: Se caracteriza por el movimiento de desplazamiento en un plano y posee dos grados de libertad.
* Articulación Esférica: Combina tres giros en tres direcciones perpendiculares en el espacio.



Figura 1. Tipos de Articulaciones.

*Eslabones*

Son aquellos que permiten al robot obtener una respuesta rápida del movimiento siendo su estructura lo más rígido y ligero posible.

*Estructura Mecánica*

La estructura mecánica de un brazo robótico, se encuentra conformada por varios eslabones unidos mediante articulaciones y esta unión logra un movimiento relativo entre cada dos eslabones consecutivos, como se observa en la Figura 2.

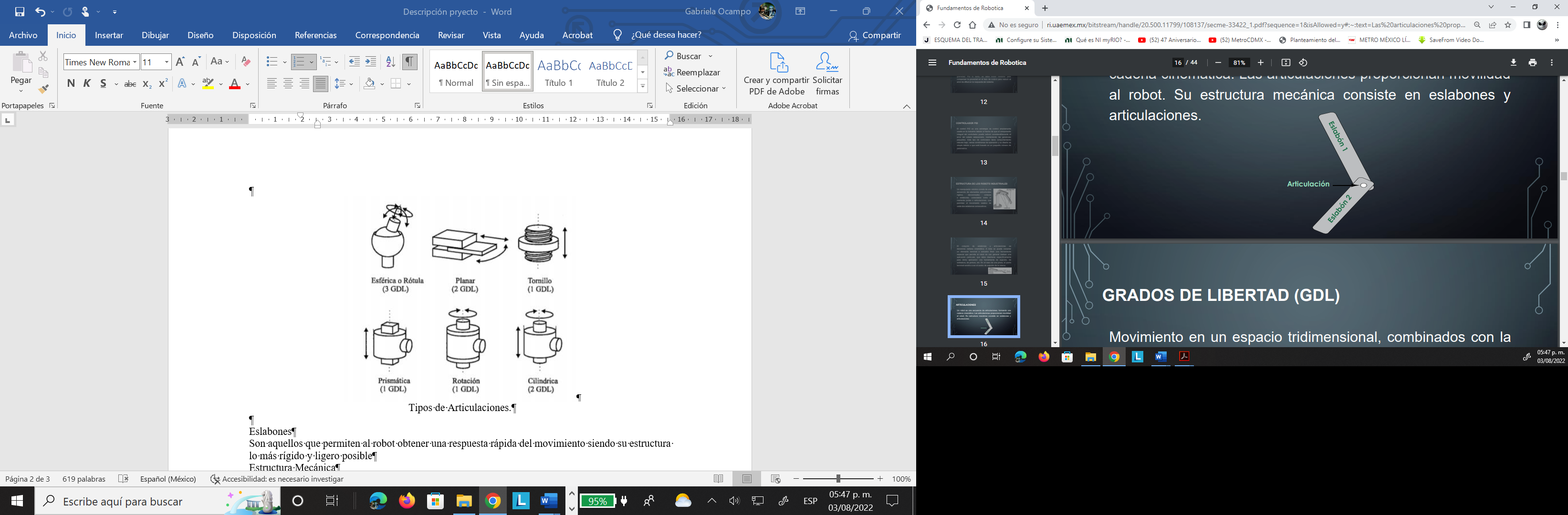


Figura 2. Estructura mecánica.

*Elementos Terminales*

El elemento o efector final de un brazo robótico, es el que se encarga de interactuar con el entorno del brazo. Básicamente es una herramienta de sujeción, que es utilizada para coger y sostener un objeto. Un elemento muy común son las pinzas de sujeción o agarre (Salazar Patin, 2015).

***Industria 4.0***

El Internet de las cosas es la interconexión de máquinas, sensores y otros dispositivos, como se observa en la Figura 3. Esta red permite que los dispositivos compartan datos y permite que tanto las máquinas como sus operadores humanos analicen continuamente los datos y tomen las medidas adecuadas para resolver problemas.

La industria 4.0 permite aprovechar la automatización inteligente que conecta a máquinas y personas, acelerando el flujo de información y análisis, eliminando la limitación de la intervención solo humana.

Además de lo anterior mencionado, la industria 4.0 no solo es control y monitorear, se debe analizar la información que arroja cada elemento conectado y aprovechar los datos que se adquieren con la finalidad de tener una mejora continua en los procesos y productos. El internet de las cosas se puede explicar como un conjunto de dispositivos u objetos físicos que se conectan a la nube y comparten información con aplicaciones a través de internet. De manera muy básica se puede enlistar de la siguiente manera.



Figura 3. Internet de las cosas.

***Servidor.***

Es un dispositivo que provee de servicios a una red bajo un modelo de cliente-servidor. Estos permiten servicios como hosting, almacenamiento de información, juegos, DNS, correo, etc. Los clientes o usuarios que se encuentran en sus equipos de trabajo solicitan servicios y un servidor se encarga de cumplir las peticiones que se le soliciten.

En un sistema IOT podremos encontrar dentro del servidor servicios como:

* Hosting
* Bases de datos
* Broker

***Hosting.***

Un servicio de hosting proporciona alojamiento a las páginas web, a las aplicaciones web y estas son de gran relevancia pues es en estas donde los usuarios van a poder interactuar con los dispositivos y la información que estos proporcionen.

***Broker***

Esta es la aplicación o sistema que permite enviar la información de los dispositivos o sensores al servidor y también se encarga de traer información a los dispositivos por parte de las aplicaciones de los usuarios o del servidor si es que existe una regla que le permita reaccionar a un evento predeterminado.

***Base de datos.***

Esta aplicación permite almacenar una gran cantidad de información de manera estructurada y relacional, como se muestra en la figura 4. Se pueden hacer consultas de acuerdo a las características de la información almacenadas y de esta manera poder obtener uno a más datos de manera precisa. Para gestionar esta información se puede utilizar sistemas como MySQL, SQL o MongoDB.



Figura 4. Servidor con base de datos.

***MQTT***

Es un protocolo de comunicación M2M que permite la comunicación entre dispositivos o sensores, por esta razón es muy utilizado en proyectos de internet de las cosas. Una de sus principales características es bajo consumo de recursos como son el ancho de banda, CPU y RAM.

Usa una topología estrella que permite al broker conectarse con hasta 10000 clientes. Se debe entender el patrón de publicación-suscripción (ver figura 5); también conocido como patrón pub-sub, un cliente que publica un mensaje se desvincula del otro cliente o clientes que reciben el mensaje. Los clientes no saben de la existencia de los otros clientes. Un cliente puede publicar mensajes de un tipo específico y solo los clientes que estén interesados en tipos específico de mensajes recibirán los mensajes publicados. El patrón de publicación-suscripción requiere un broker; también conocido como servidor. Todos los clientes establecen una conexión con el servidor. El cliente que enviar un mensaje a través del broker se conoce como publicador. El servidor filtra los mensajes entrantes y los distribuye a los clientes que están interesados en el tipo de mensajes recibidos. Los clientes que se registran en el servidor como interesados en tipos específicos de mensajes se conocen como suscriptores. Por eso; tanto los clientes como los suscriptores establecen una conexión con el bróker.

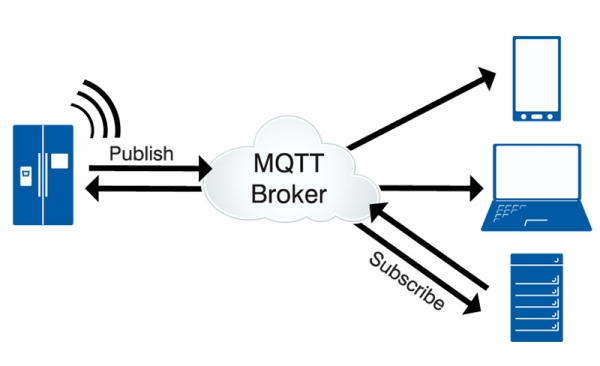


Figura 5. Protocolo de comunicación MQTT.

***Node-RED***

Es una herramienta que se programa por medio de bloques y sirve para conectar dispositivos físicos, API y servicios en línea, como se muestra en la figura 6. El lenguaje de programación que utiliza Node-RED es JavaScript.

Node-Red está basada en Node.js, esto permite que aproveche al máximo su modelo basado en eventos y sin bloqueo. Gracias a esto Node-RED es ideal para ejecutarse en el borde de la red en hardware de bajo costo como la Raspbery Pi, así como en la nube.

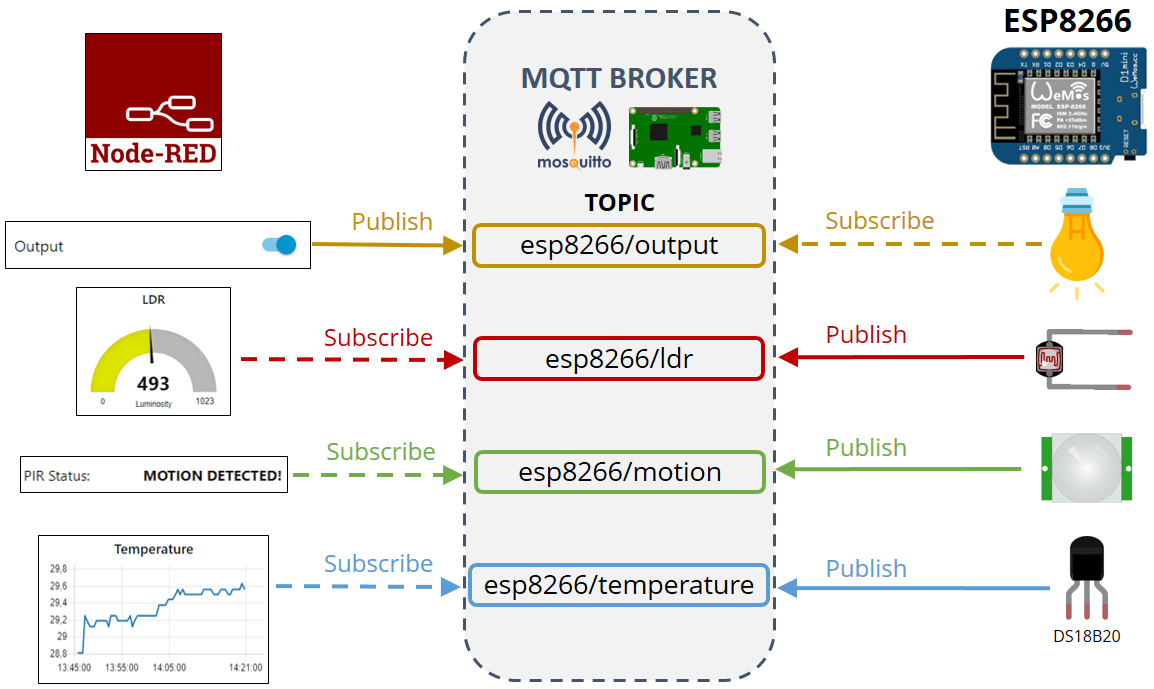


Figura 6. Node-Red.

**Desarrollo.**

**Diseño del brazo robótico en el software CAD**

El diseño asistido por computadora (CAD) se refiere al uso de computadoras para crear, modificar, optimizar y analizar un diseño. Y el software que permite este tipo de diseño en una computadora se llama software CAD.

El software CAD viene con una interfaz fácil de usar y varias herramientas y funciones para que el proceso de diseño sea perfecto y preciso. Puede crear modelos bidimensionales y tridimensionales de componentes físicos e imprimir los diseños en una página para llevarla a los siguientes pasos de producción.

El software CAD también facilita el flujo del proceso de diseño desde el diseño hasta la fabricación y simula el movimiento de un modelo en 3D. El software se utiliza en todas las industrias, incluida la ingeniería

En esta ocasión el diseño del brazo robot se realizó utilizando el software de diseño paramétrico SolidWorks (Dassault Systèmes, 2017). Mediante este software, en el modo “Pieza” se crearon todos los elementos del brazo.

En la Figura 7 y 8, se puede apreciar el diseño de los dos eslabones con los que cuenta el brazo robot así como la base.

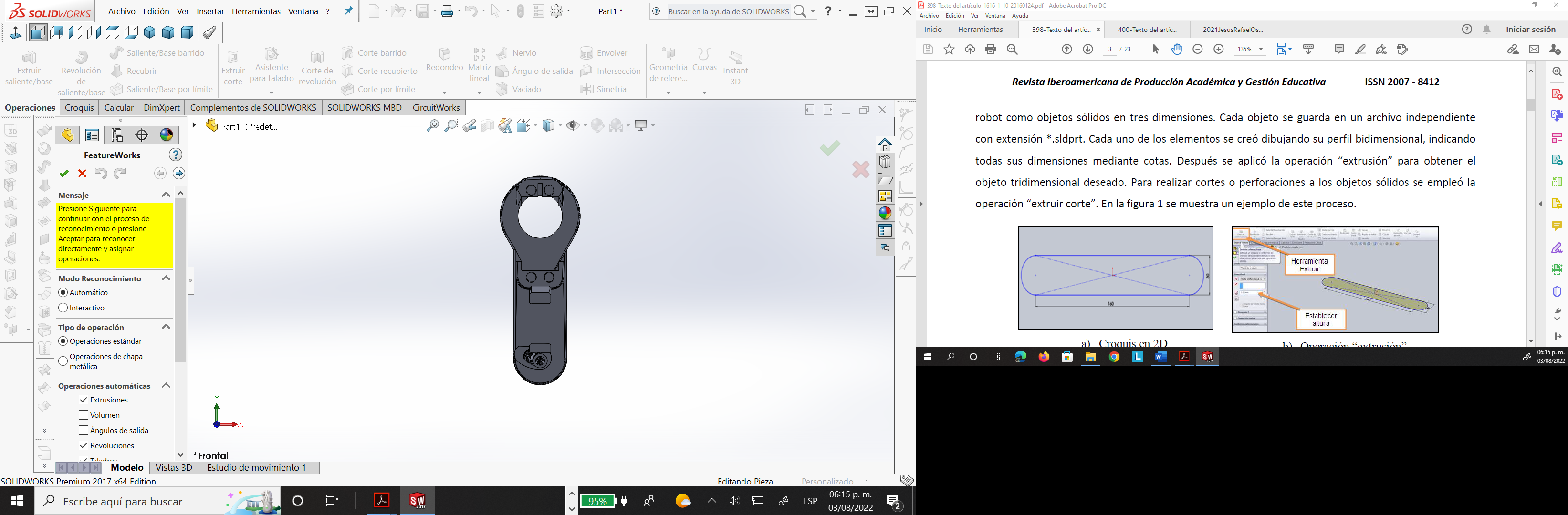
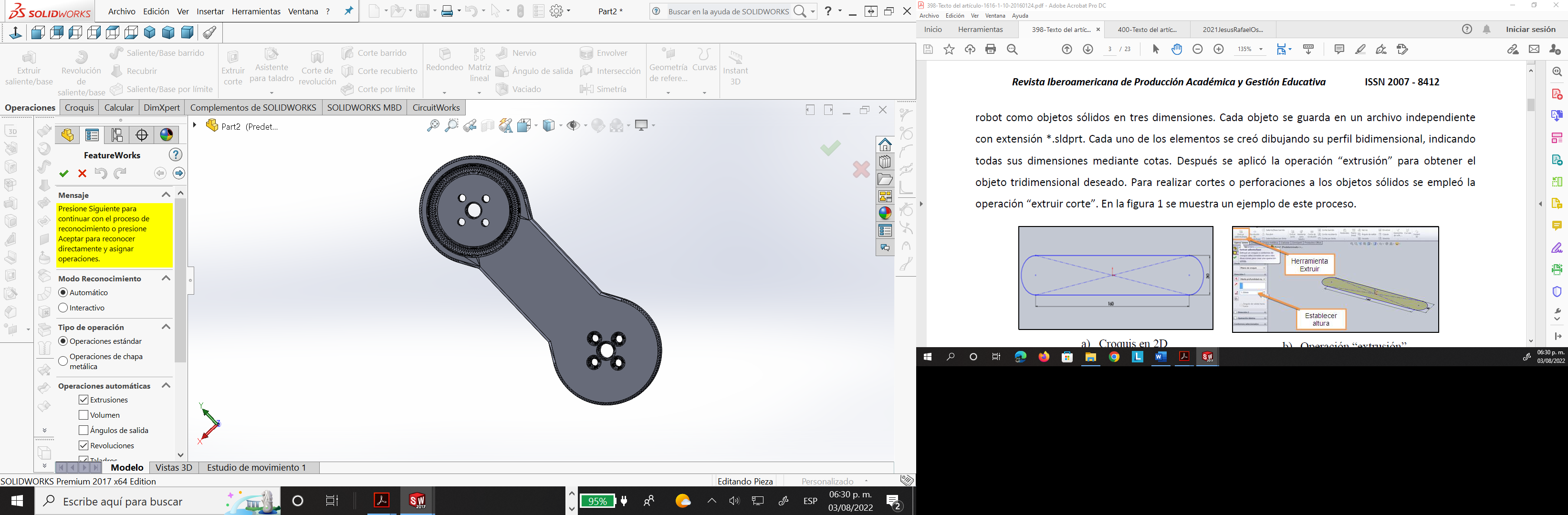


Figura 7. Eslabón L1 y eslabón L2.

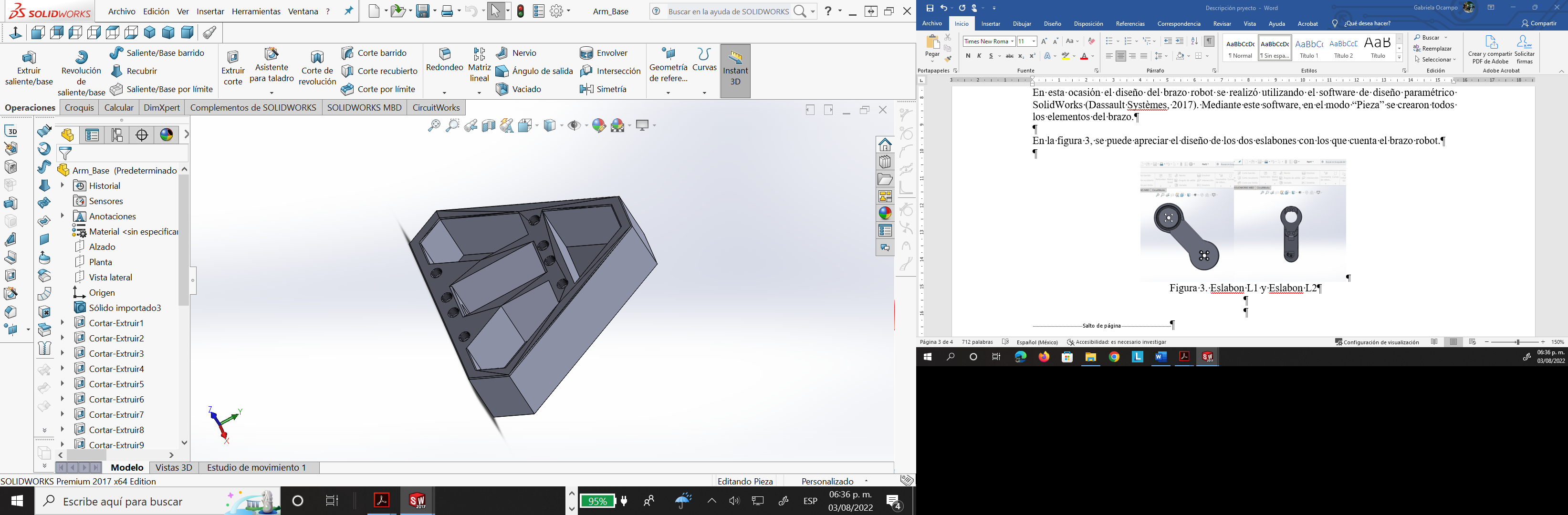


Figura 8. Base del robot.

En la Figura 9 se puede apreciar el ensamble que se realizó en el software CAD SolidWorks.

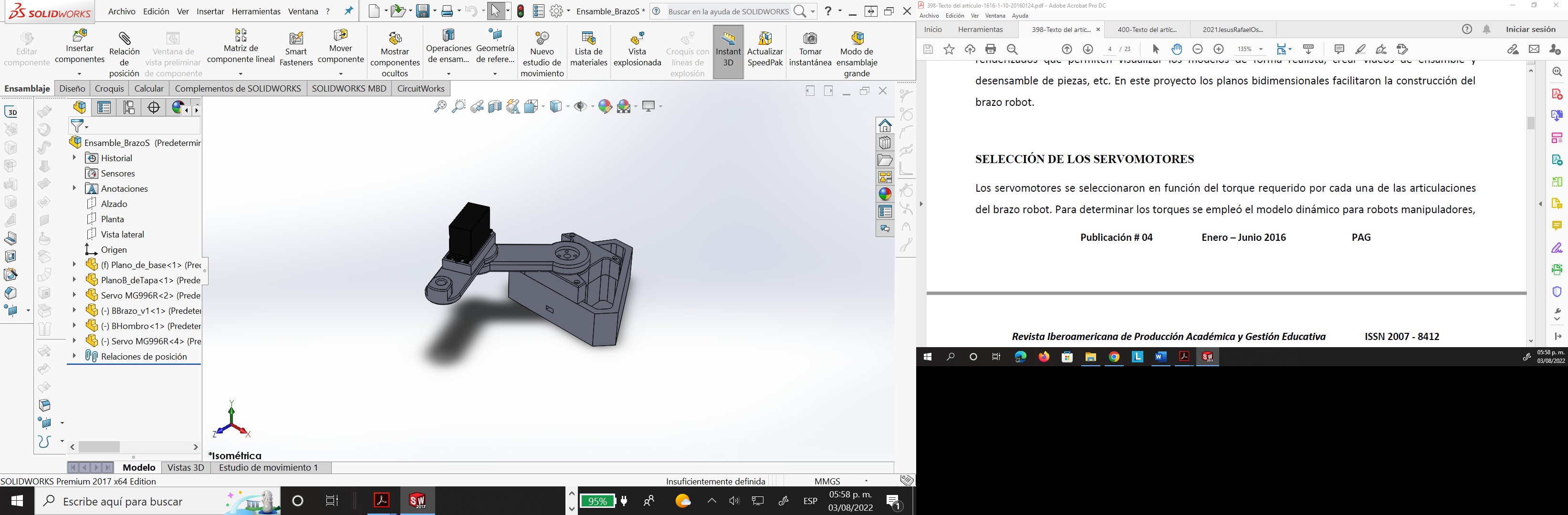


Figura 9. Componentes ensamblados.

***Modelación matemática la cinemática inversa del brazo robótico.***

Para entender cómo se realizó el modelo matemático de la cinemática inversa, se explicara primero que es la cinemática directa y posteriormente la cinemática inversa.

***Cinemática directa de un brazo robótico.***

La cinemática directa estudia el movimiento de las estructuras mecánicas del brazo robótico.

Un brazo robótico puede tener varios grados de libertad. Para cada eslabón del brazo robótico se le asigna un sistema de coordenadas, con el origen sobre la articulación que controla el movimiento, como se muestra en la figura 10. El movimiento puede ser rotatorio o traslatorio.

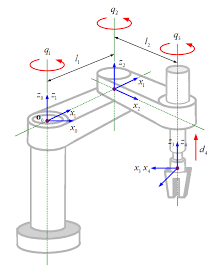


Figura 10. Sistema de coordenadas para un brazo robótico.

***Cinemática inversa de un brazo robótico.***

Si se desea mover cada una de las articulaciones del brazo robótico para llegar a una cierta posición deseada (ver Figura 11), se deben obtener expresiones para las variables de eslabón (esto es cuando el movimiento es rotatorio y d cuanto el movimiento es traslatorio) en función de la posición deseada. Las expresiones resultantes son las ecuaciones cinemáticas inversas.

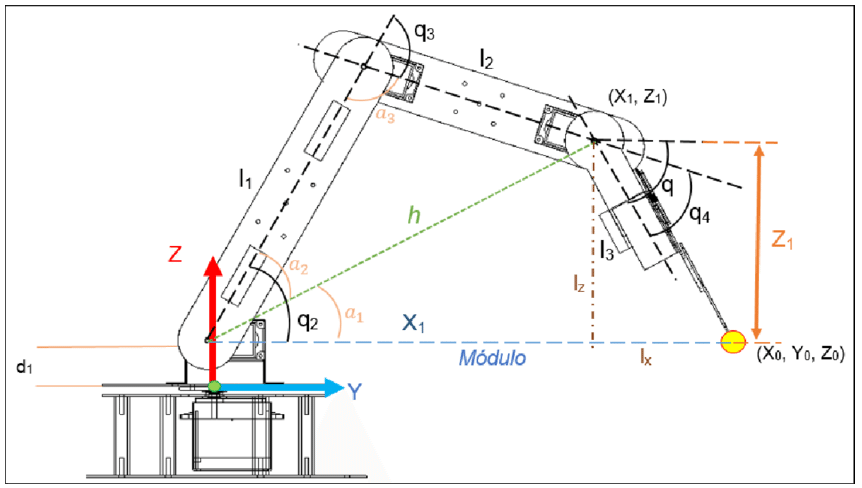


Figura 11. Posición deseada (Xo, Yo, Zo), para mover el brazo robótico.

La cinemática inversa del brazo robótico se programó de acuerdo a la figura 12. Como se puede observar primero el usuario debe definir las coordenadas *xi, yi, xf* y *yf* para generar la trayectoria deseada para que el brazo se mueva, después aplicamos las ecuaciones de la cinemática inversa para calcular los ángulos que se enviaran a cada servo que se encuentran en las articulaciones del robot.

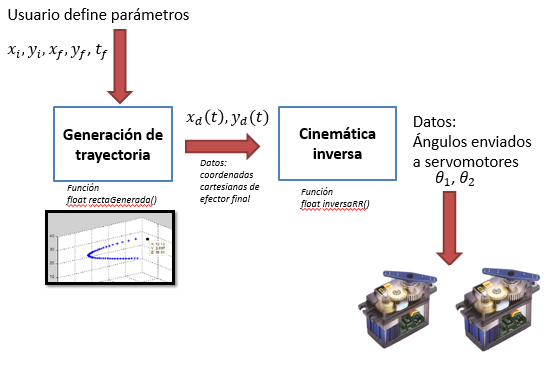


Figura 12. Diagrama del programa de la cinemática inversa del brazo robótico.

El código de este programa se encuentra en el Anexo A.

**Dispositivos electrónicos para el IoT.**

***Sensores de posición: Potenciómetros***

Un potenciómetro es un transductor entre la posición de un objeto, ya sea lineal o angular, y un cambio de resistencia. Este tipo de elementos resistivos se utiliza normalmente con un voltaje en corriente directa. Constan de tres terminales, una en cada extremo del material, y una tercera terminal que recorre el cuerpo del elemento resistivo, de tal manera que la resistencia entre la terminal móvil y cada una de las terminales fijas varia cuando el elemento móvil cambia de posición; así, cuando una resistencia disminuye, la otra necesariamente aumenta. El circuito equivalente se puede observar en la Figura 13.

En este trabajo se utilizaron dos potenciómetros para medir la posición de cada articulación del brazo robótico.

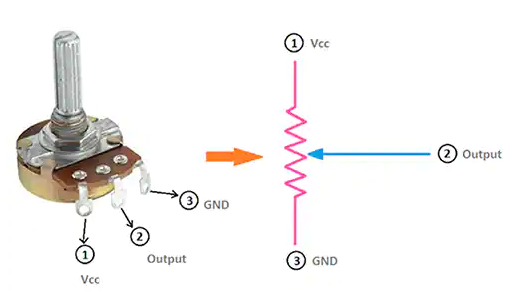


Figura 13. Circuito de un potenciómetro.

***ESP32.***

La ESP32 tiene un procesador de doble núcleo que puede funcionar con frecuencias de hasta 240 MHz, con 512 KB de memoria SRAM y opciones de conectividad ampliadas al incorporar en el propio chip el hardware para comunicación por Ethernet, bluetooth y Wifi. La figura 14 muestra la estructura de la ESP32

Para este trabajo la ESP32 se conectó a Wifi para enviar y recibir mensajes MQTT.

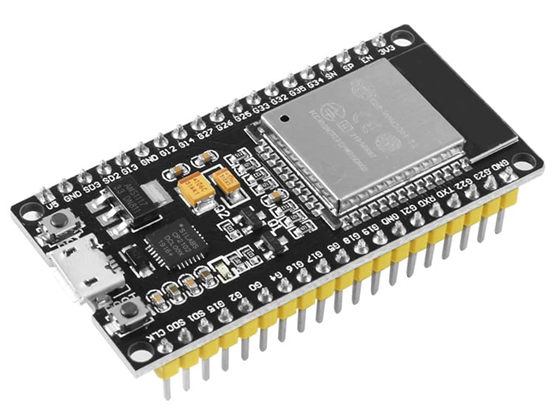


Figura 14. ESP32

**Comunicación a distancia con el protocolo de comunicación MQTT.**

Para lograr la comunicación con MQTT, se realizó un programa en la ESP32, el cual se conecta a Wifi, a un bróker MQTT.

Este programa detecta la recepción de las coordenadas *xi, yi, xf y yf* para generar la trayectoria deseada y posteriormente determina los ángulos que se enviaran a cada uno de los servos que se encuentran en las articulaciones del brazo robótico. La Figura 15 muestra como son enviados los datos por MQTT.

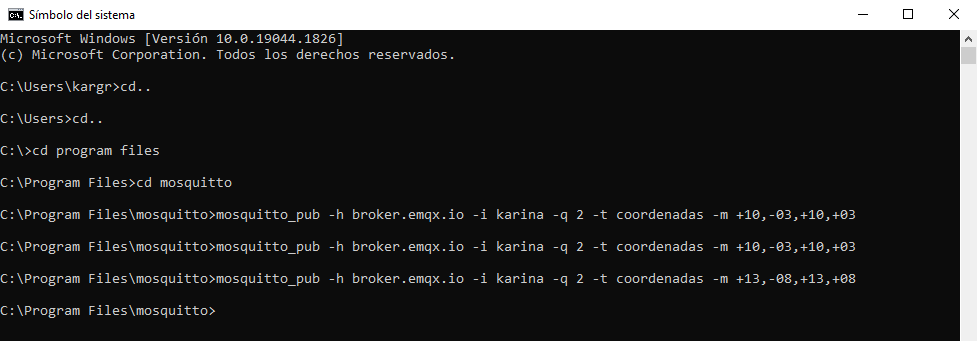


Figura 15. Recepción de coordenadas con MQTT

También este programa envía las posiciones de cada articulación, como se muestra en las figuras 16 y 17. Esta posición es medida por los potenciómetros que se encuentran en cada una de las articulaciones.

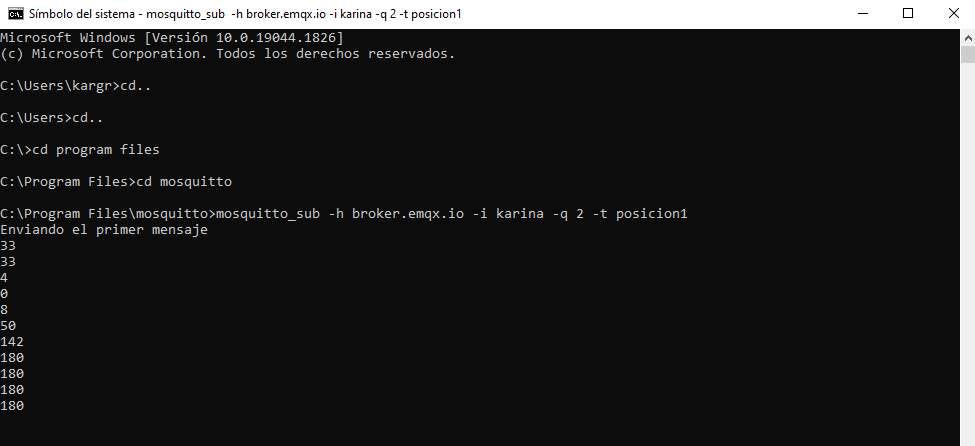


Figura 16. Envío de posición 1 con MQTT.

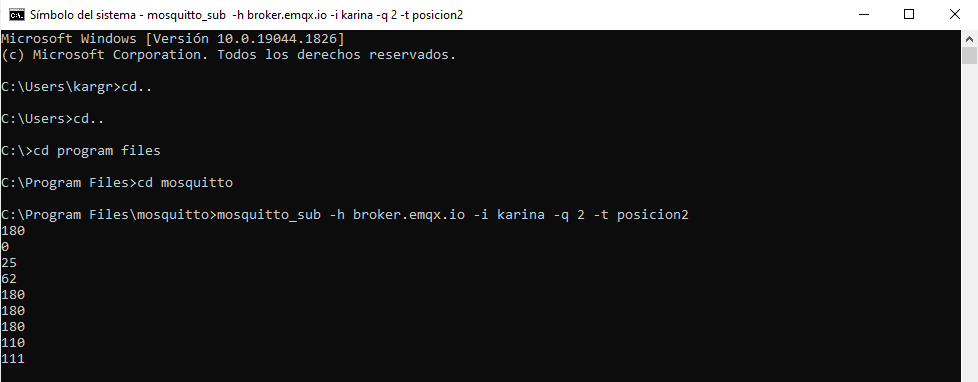


Figura 17. Envío de posición 2 con MQTT.

El código de este programa se encuentra en el Anexo B.

**Comunicación a distancia con el protocolo de comunicación MQTT y Node-Red.**

Para poder hacer la comunicación usando Node-red, se debe de hacer un flow. Este flow consistió en tomar un nodo de MQTT in y conectarlo a otro nodo de gauge para el envió de la posición 1. Este mismo procedimiento se repitió para la posición dos. Por otra parte para la recepción de las coordenadas se tomó un nodo de text input y se conectó a otro nodo de template y este nodo se conectó a otro nodo de MQTT out. En la figura 18 se puede observar este flow

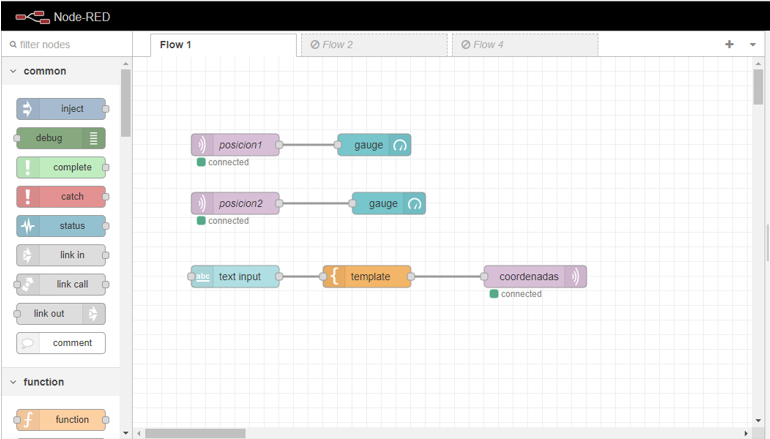


Figura 18. Flow para el envío y recepción de información en Node-Red

Por último se puede observar en la Figura 19 las gráficas de las posiciones de las articulaciones del brazo robótico y el envío de las coordenadas en Node-Red.

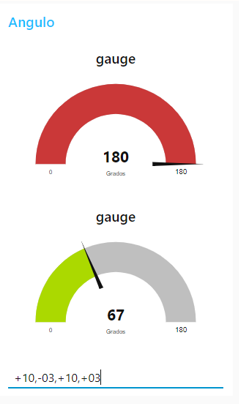
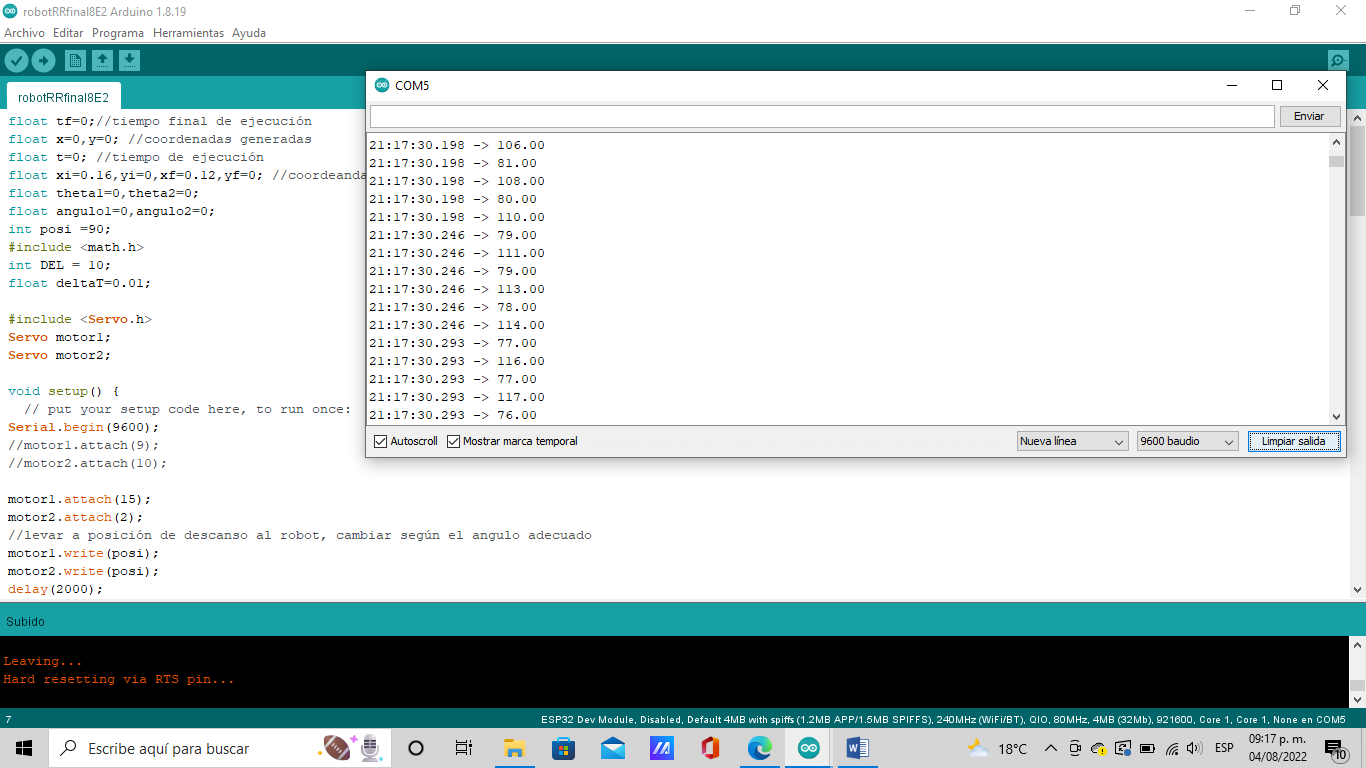
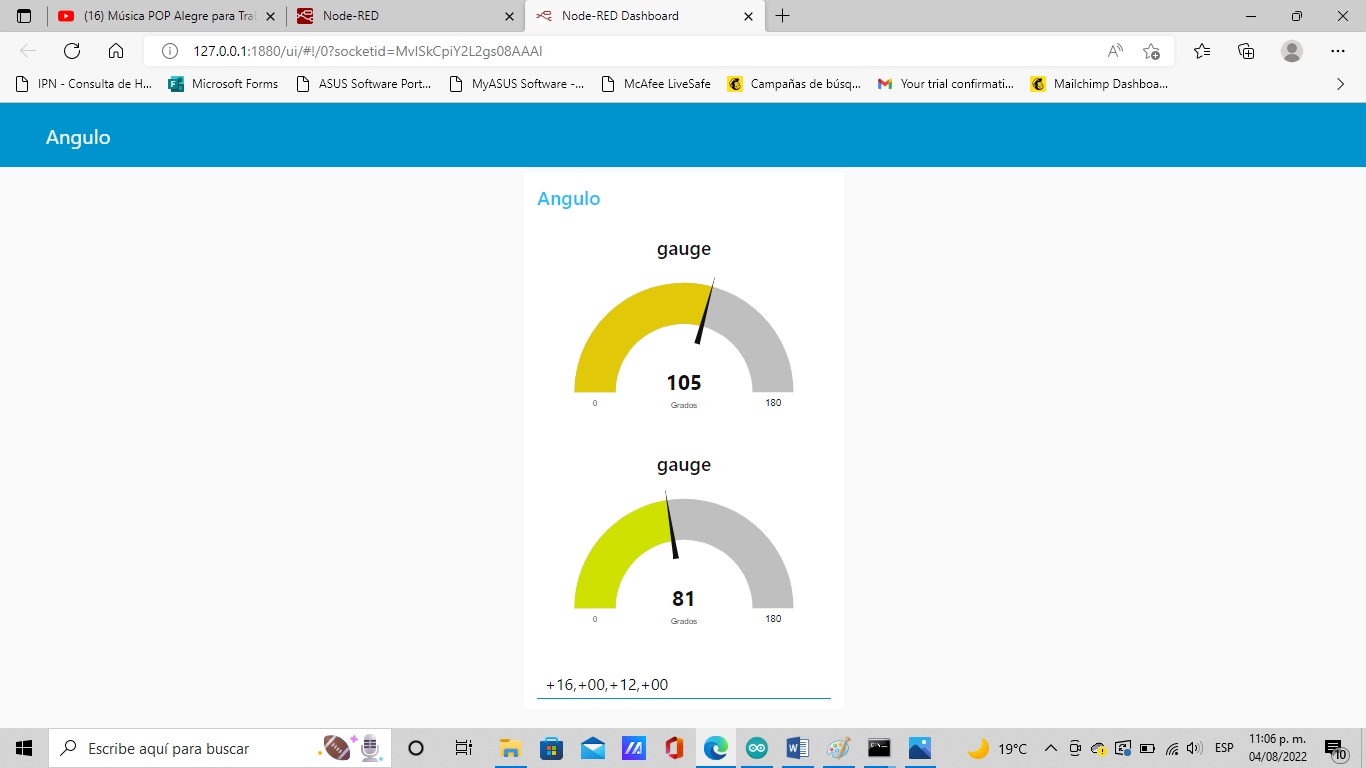
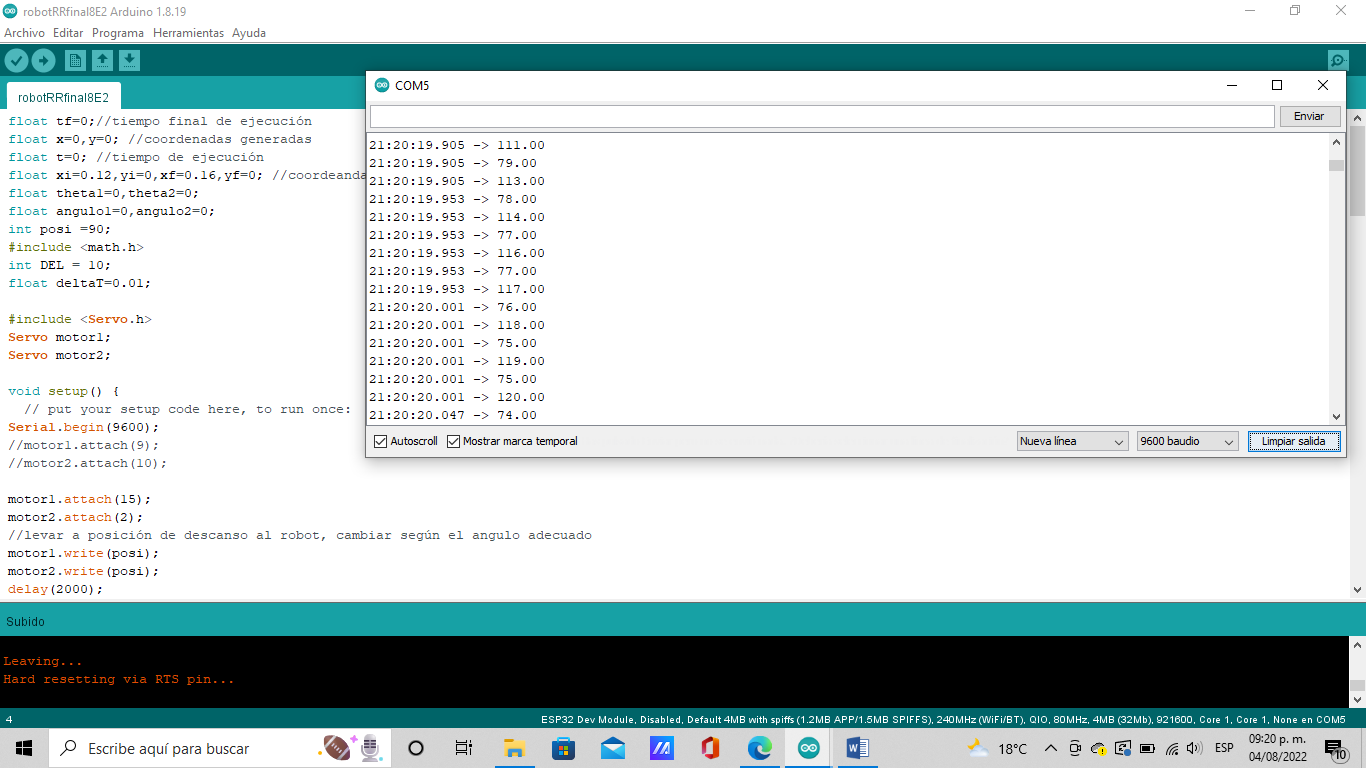
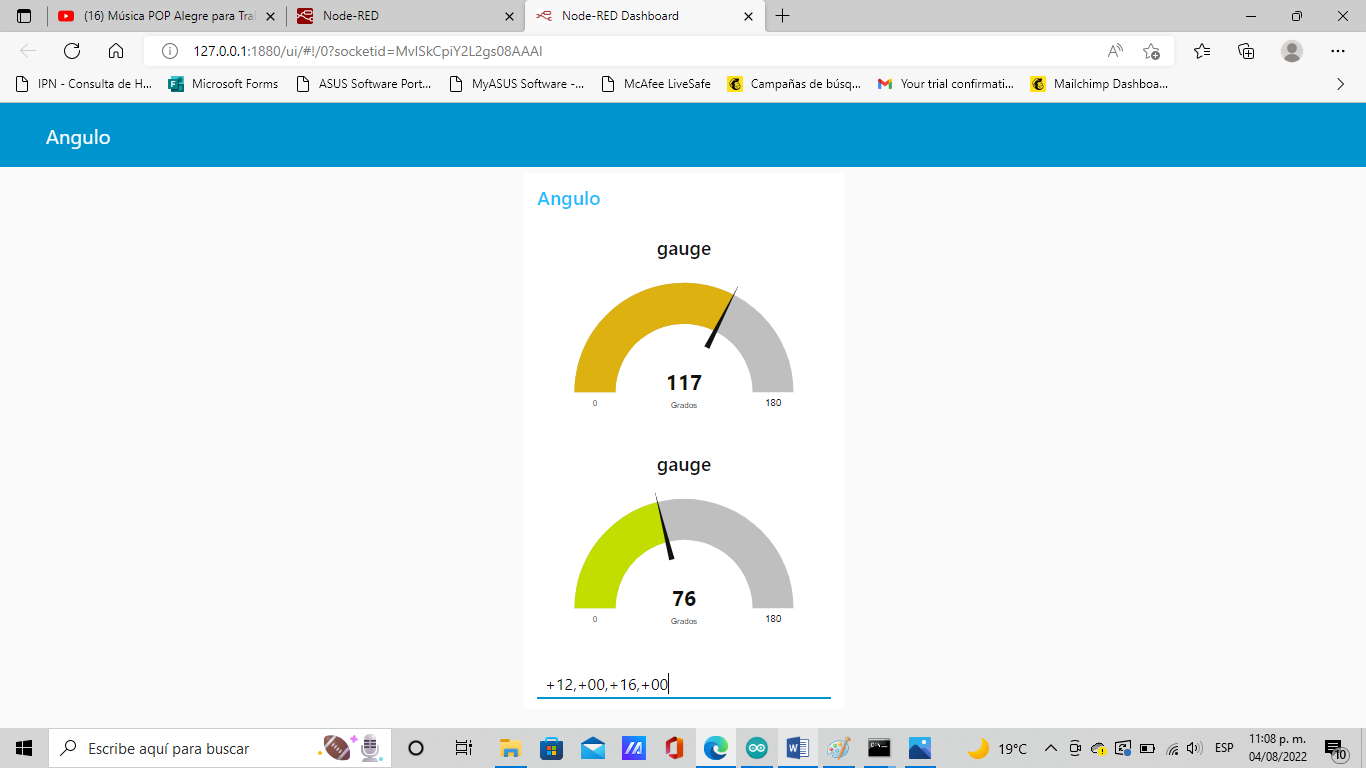


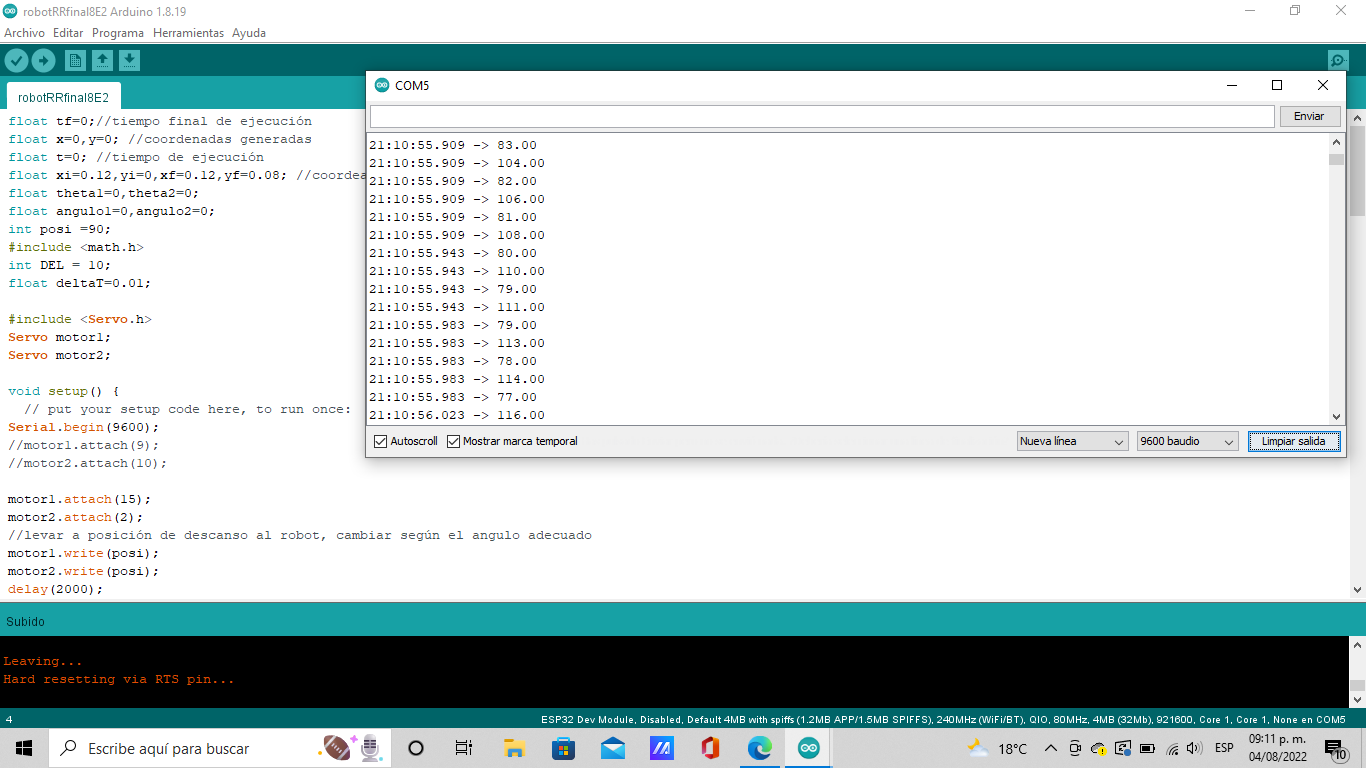
Figura 19. Gráficas de posición del brazo robótico y envío de las coordenadas.

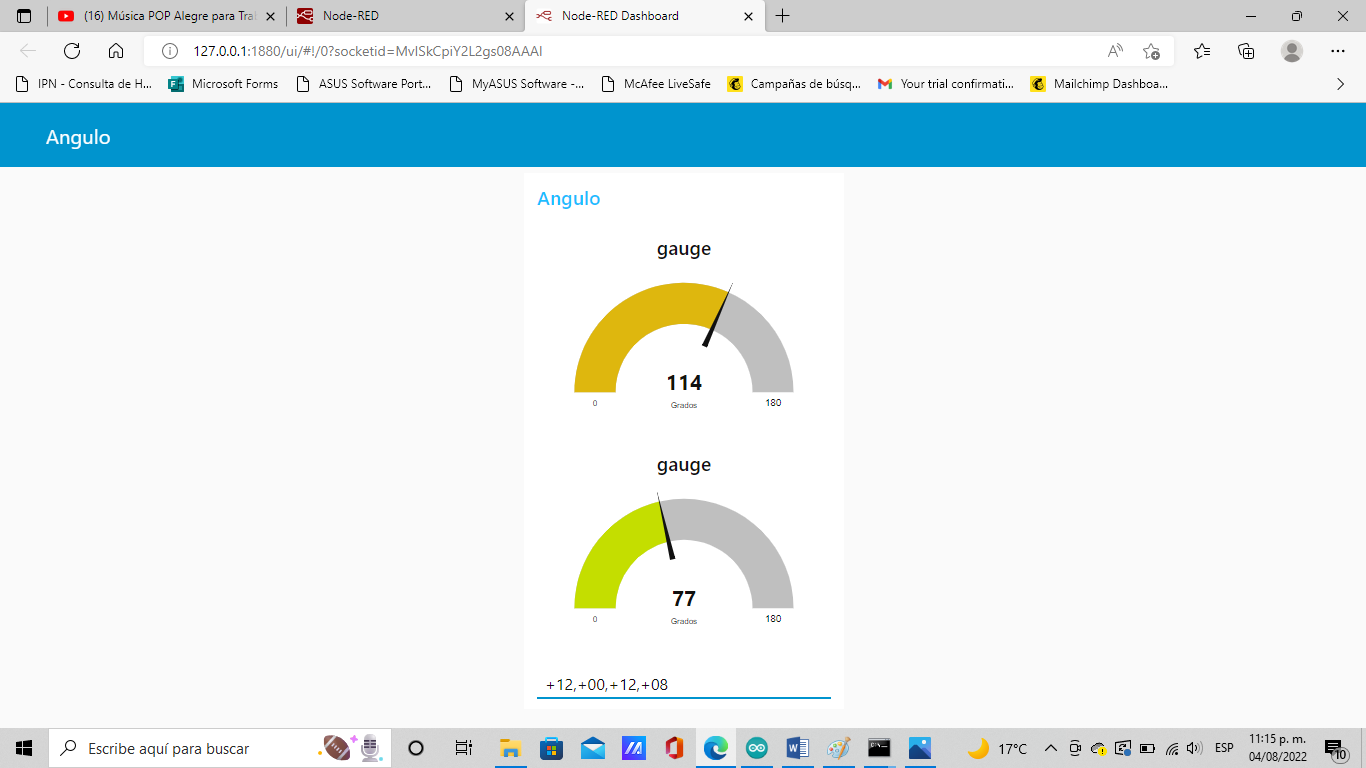


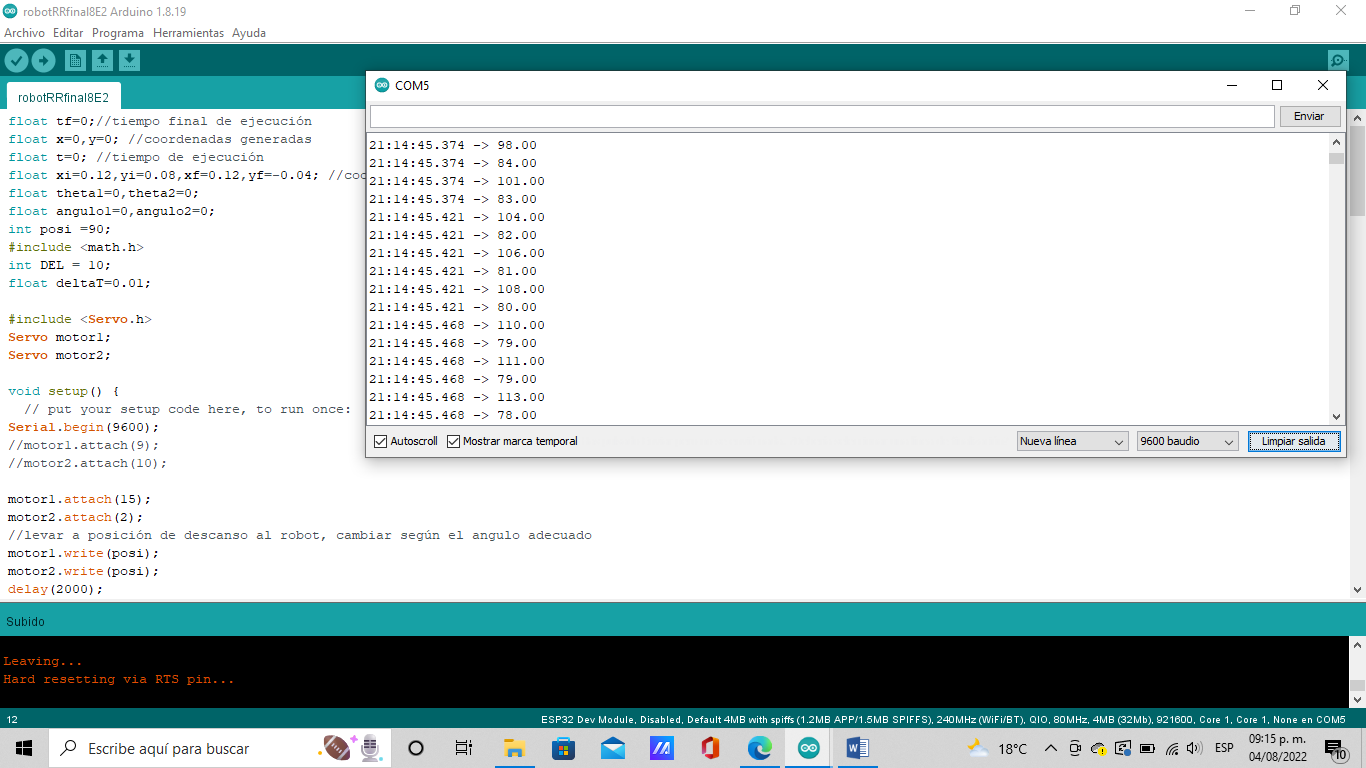


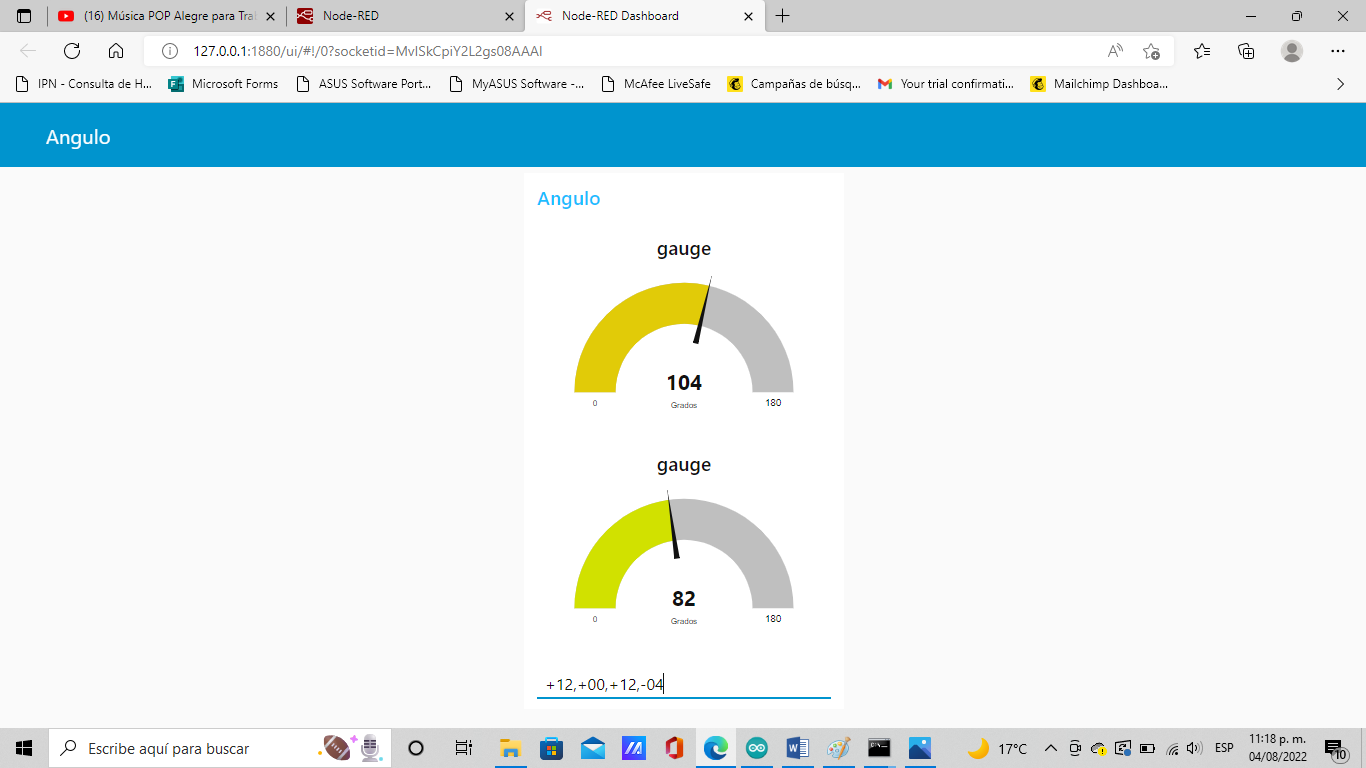












**Conclusiones y trabajo futuro.**

* Se logró el objetivo principal gracias a las herramientas vistas en el diplomado.
* Mejoras como más rutinas, interfaz con otras plataformas, menú, etc.
* A futuro estudiar la dinámica con los efectos de discretización asociados a la red.
* Extender a mas robots de diversos tipos (Rover, drones, bípedos) o más grados de libertad.

**ANEXO A. Código del programa de la cinemática inversa del brazo robótico.**

float tf=0;//tiempo final de ejecución

float x=0,y=0; //coordenadas generadas

float t=0; //tiempo de ejecución

float xi=13,yi=-8,xf=13,yf=8; //coordeandas iniciales y finales

float theta1=0,theta2=0;

float angulo1=0,angulo2=0;

int posi =90;

#include <math.h>

int DEL = 10;

float deltaT=0.01;

#include <Servo.h>

Servo motor1;

Servo motor2;

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

//motor1.attach(9);

//motor2.attach(10);

motor1.attach(15);

motor2.attach(2);

//levar a posición de descanso al robot, cambiar según el angulo adecuado

motor1.write(posi);

motor2.write(posi);

delay(2000);

tf=1;

for(t=0;t<=tf;t+=deltaT){

rectaGenerada(tf,t,0.16,0.0,0.12,0.0);

inversaRR(x,y);

angulo1=posi+theta1;

angulo2=posi+theta2;

//motor1.write(posi+theta1);

//motor2.write(posi+theta2);

Serial.println(angulo1);

Serial.println(angulo2);

motor1.write(angulo1);

motor2.write(angulo2);

delay(DEL);}

for(t=0;t<=tf;t+=deltaT){

rectaGenerada(tf,t,0.12,0.0,0.16,0.0);

inversaRR(x,y);

angulo1=posi+theta1;

angulo2=posi+theta2;

//motor1.write(posi+theta1);

//motor2.write(posi+theta2);

Serial.println(angulo1);

Serial.println(angulo2);

motor1.write(angulo1);

motor2.write(angulo2);

delay(DEL);}

}

void loop() {

}

float rectaGenerada(float tf,float t,float xi,float yi,float xf,float yf){

float vx=(xf-xi)/tf;

float vy=(yf-yi)/tf;

x=xi+vx\*t;

y=yi+vy\*t;

return x,y;

}

float inversaRR(float x,float y){

float L1=0.08;

float L2=0.08;

theta2=acos((pow(x,2)+pow(y,2)-pow(L1,2)-pow(L2,2))/(2\*L1\*L2));

theta1=atan(y/x)-atan((L2\*sin(theta2))/(L1+L2\*cos(theta2)));

//conversion a grados par aservos

theta1=round(180\*(theta1/3.14));

theta2=round(180\*(theta2/3.14));

return theta1,theta2;

}

**ANEXO B. Código del programa de envío y recepción de información con MQTT en la ESP32.**

#include <WiFi.h>

#include <PubSubClient.h>

#include <Servo.h>

#include <math.h>

Servo motor1;

Servo motor2;

// Update these with values suitable for your network.

//const char\* ssid = "¿Quién va a reprobar?";

//const char\* password = "todoelgrupo";

//const char\* mqtt\_server = "broker.emqx.io";

const char\* ssid = "INFINITUM6211\_2.4";

const char\* password = "1ciGnkCo4e";

//const char\* mqtt\_server = "127.0.0.1";

const char\* mqtt\_server = "broker.emqx.io";

WiFiClient espClient;

PubSubClient client(espClient);

unsigned long lastMsg = 0;

#define MSG\_BUFFER\_SIZE (50)

//#define BUILTIN\_LED

char msg1[MSG\_BUFFER\_SIZE];

char msg2[MSG\_BUFFER\_SIZE];

int value = 0;

int pot1 = 0;

int pot2 = 0;

int pot3 = 0;

int i;

int o;

int p;

int q;

int r;

float v1=0;

float v2=0;

float v3=0;

int angulo1=0;

int angulo2=0;

int angulo3=0;

byte angulos;

float tf=0;//tiempo final de ejecución

float x=0,y=0; //coordenadas generadas

float xi,yi,xf,yf; //coordeandas iniciales y finales

float t=0; //tiempo de ejecución

float theta1=0,theta2=0;

int posi =90;

int DEL = 10;

float deltaT=0.01;

void setup\_wifi()

{

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

randomSeed(micros());

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

int callback1(char\* topic, byte\* payload, unsigned int length) //resepción de de datos desde el broker

{

Serial.print("Message arrived [");

Serial.print(topic);

Serial.println("] ");

for (int i = 0; i < length; i++)

{

Serial.println((char)payload[i]);

}

int a=payload[1]-48;

int b=payload[2]-48;

int c=payload[5]-48;

int d=payload[6]-48;

int e=payload[9]-48;

int f=payload[10]-48;

int g=payload[13]-48;

int h=payload[14]-48;

int j=payload[0];

int k=payload[4];

int l=payload[8];

int m=payload[12];

if (j==45)

{

o=(j-45)-1;

}

if (j==43)

{

o=(j-43)+1;

}

if (k==45)

{

p=(k-45)-1;

}

if (k==43)

{

p=(k-43)+1;

}

if (l==45)

{

q=(l-45)-1;

}

if (l==43)

{

q=(l-43)+1;

}

if (m==45)

{

r=(m-45)-1;

}

if (m==43)

{

r=(m-43)+1;

}

xi=(((a\*10)+b)\*o)/100;

yi=(((c\*10)+d)\*p)/100;

xf=(((e\*10)+f)\*q)/100;

yf=(((g\*10)+h)\*r)/100;

Serial.println(a);

Serial.println(b);

Serial.println(c);

Serial.println(d);

Serial.println(xi);

Serial.println(yi);

Serial.println(xf);

Serial.println(yf);

return xi,yi,xf,yf;

}

void reconnect()

{

// Loop until we're reconnected

while (!client.connected())

{

Serial.print("Attempting MQTT connection...");

// Create a random client ID

String clientId = "ESP8266Client-";

clientId += String(random(0xffff), HEX);

// Attempt to connect

if (client.connect(clientId.c\_str()))

{

Serial.println("connected");

// Once connected, publish an announcement...

client.publish("posicion1", "Enviando el primer mensaje");

client.publish("posicion2", "Enviando el primer mensaje");

// ... and resubscribe

client.subscribe("coordenadas");

}

else

{

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying

delay(5000);

}

}

}

void setup()

{

//pinMode(5, OUTPUT); // Initialize the BUILTIN\_LED pin as an output

Serial.begin(115200);

setup\_wifi();

client.setServer(mqtt\_server, 1883);

client.setCallback(callback1);

motor1.attach(15);

motor2.attach(2);

//levar a posición de descanso al robot, cambiar según el angulo adecuado

motor1.write(posi);

motor2.write(posi);

delay(2000);

tf=1;

for(t=0;t<=tf;t+=deltaT)

{

rectaGenerada(tf,t,0.16,0.0,0.12,0.0);

inversaRR(x,y);

motor1.write(posi+theta1);

motor2.write(posi+theta2);

delay(DEL);

}

for(t=0;t<=tf;t+=deltaT)

{

rectaGenerada(tf,t,0.12,0.0,0.16,0.0);

inversaRR(x,y);

motor1.write(posi+theta1);

motor2.write(posi+theta2);

delay(DEL);

}

}

void loop()

{

if (!client.connected())

{

reconnect();

}

client.loop();

//Para el envio de datos

pot1 = analogRead(36);

pot2 = analogRead(39);

v1=pot1\*(3.3/4095);

v2=pot2\*(3.3/4095);

angulo1=(v1\*180)/3.3;

angulo2=(v2\*180)/3.3;

unsigned long now = millis();

if (now - lastMsg > 2000)

{

lastMsg = now;

++value;

//snprintf (msg1, MSG\_BUFFER\_SIZE, "%ld, %ld, %ld", angulo1, angulo2, angulo3);

snprintf (msg1, MSG\_BUFFER\_SIZE, "%ld", angulo1);

snprintf (msg2, MSG\_BUFFER\_SIZE, "%ld", angulo2);

client.publish("posicion1", msg1);

client.publish("posicion2", msg2);

}

}

float rectaGenerada(float tf,float t,float xi,float yi,float xf,float yf)

{

float vx=(xf-xi)/tf;

float vy=(yf-yi)/tf;

x=xi+vx\*t;

y=yi+vy\*t;

return x,y;

}

float inversaRR(float x,float y)

{

float L1=0.08;

float L2=0.08;

theta2=acos((pow(x,2)+pow(y,2)-pow(L1,2)-pow(L2,2))/(2\*L1\*L2));

theta1=atan(y/x)-atan((L2\*sin(theta2))/(L1+L2\*cos(theta2)));

//conversion a grados par aservos

theta1=round(180\*(theta1/3.14));

theta2=round(180\*(theta2/3.14));

return theta1,theta2;

}