INFORME

Alumnos:

Cristian Nahuel Castañares

DNI: 42339695

MAIL: crismust99@gmail.com

Materia:

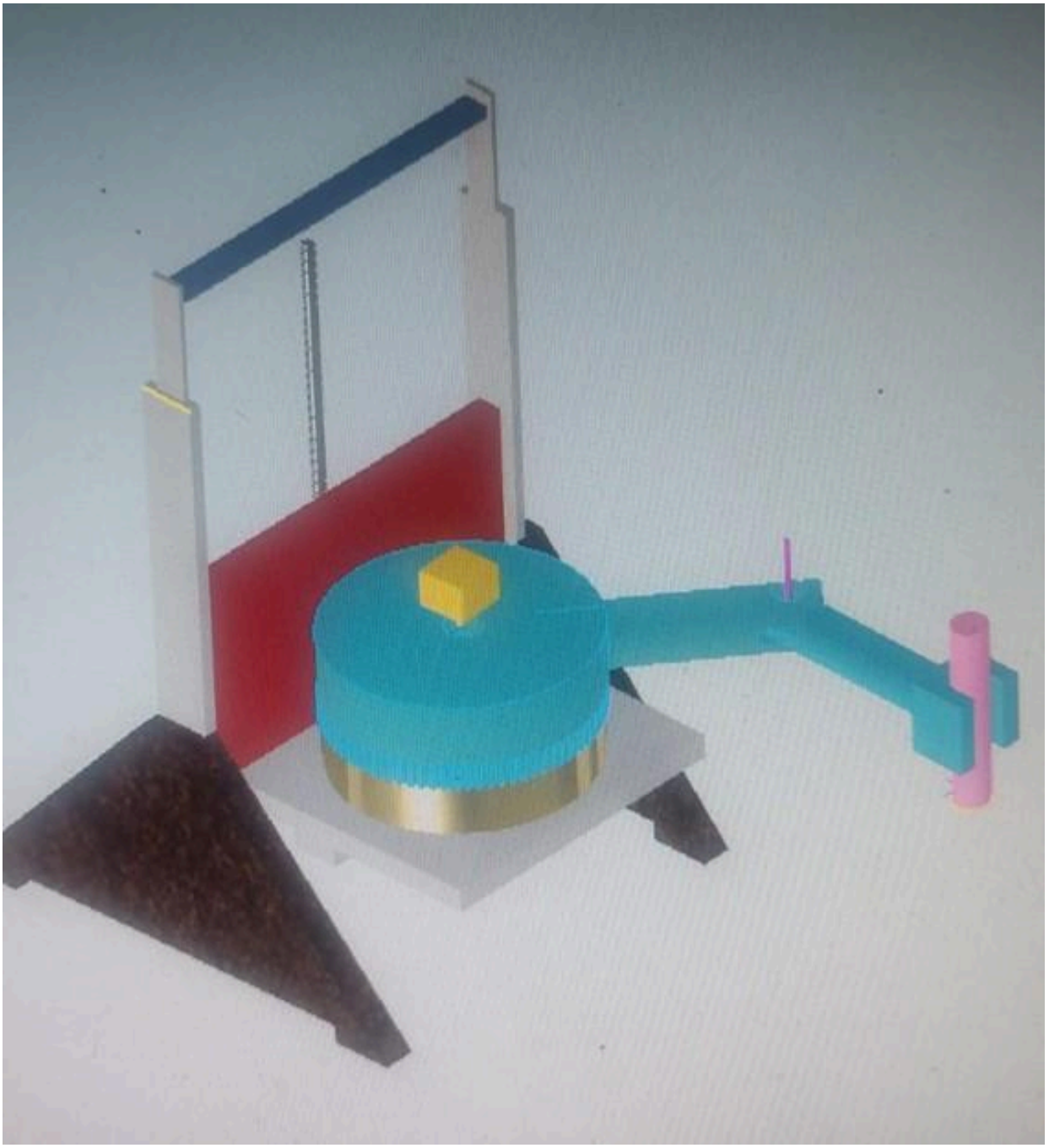
Proyecto Final de Ingeniería Mecatrónica

Carrera:

Ingeniería Mecatrónica

Observaciones

Armado de brazo robótico para imprimir en 3D utilizando plástico fundido



Índice

1. Presentación

1.1 Introducción

Se realizará un brazo robótico de 3 grados de libertad, uno prismático y dos rotativos, con una distribución tipo SCARA – PRR.
Se controlarán los motores paso a paso correspondientes a cada grado de libertad, a través de un Arduino Uno junto con controladores Pololu A4988.
Se agregará una función de conexión a través de WIFI con la integración de un ESP32 Cam, que se intercomunica con el Arduino Uno, con el cual se podrá enviar el archivo en código G (.GCODE) de la pieza a imprimir, y comenzará automáticamente. La comunicación entre la computadora donde se encuentra el código G y el ESP32 de la impresora, se realizará a través de la pagina web. Con un desarrollo mayor, se podría controlar a distancias grandes a través de páginas web, por ejemplo.
En una futura aplicación se podría reemplazar la base de impresión por una cinta que soporte la temperatura de extrusión a fin de poder contar con un nivel de producción en serie superior a la mayoría de las máquinas disponibles en el mercado.

1.2 Objetivos

- En este proyecto se busca diseñar un sistema de impresión 3D diferente a los modelos más comunes del mercado, como la impresora de 3 ejes o las de un eje infinito. Además, tendrá mayor capacidad de impresión en cuanto al tamaño de las impresiones que pueda realizar, ya que al ser un instrumento bastante utilizado en la actualidad, nace una necesidad de imprimir piezas más grandes de una sola vez.
- También se busca mejorar la relación humano – impresora a través de una comunicación online, ya sea para mostrar el avance de la impresión como también adjuntarle un nuevo trabajo sin necesidad de conectarla a una PC o a través de una tarjeta de memoria compatible.
- Otro de los objetivos es poder construir una impresora que conlleve un menor gasto de fabricación para así competir con demás impresoras en el mercado que posean un precio mayor por características iguales o inferiores.

1.3 Alcance

- Para el alcance de este proyecto, se tendrán en cuentas las siguientes modificaciones al proyecto inicialmente descripto:
- Se reemplazará el extrusor normalmente utilizado en impresoras 3D por un lápiz que cumple la misma funcionalidad pero con la ventaja de ser mucho más sencillo de utilizar.
- No se tendrá en cuenta el eje infinito para producción en serie, y se trabajará solamente con el área de impresión física del robot.
- La comunicación entre la computadora y la impresora será a través de wifi en la cual ambos dispositivos deberán estar conectado a la misma red.

1.4 Resumen

Se elaborará un brazo robótico de 3 grados de libertad, con una configuración de 1 eje prismático que se utilizara para el movimiento vertical de la extrusora, y 2 ejes rotativos en formato SCARA.
La ventaja de esta configuración es que nos da un área de impresión de 180°, un alcance de 41 cm (de base pudiendo modificar según características) y una altura de 30 cm aproximadamente. En total seria una area de impresion de [Calculo del area de impresion].
Las piezas a utilizar serán de PLA, ya que se diseñaron e imprimieron en 3D, madera, para los soportes y componentes que requieran una resistencia mayor, y por último se integran los componentes electrónicos.
El funcionamiento se basa en el envío del archivo con extensión .gcode que se obtiene en cualquier laminador del mercado (como por ejemplo Cura) a través de una página web, y que pueda imprimirse automáticamente sin necesitar de tener acceso físico a la máquina. También contará con una cámara a la que se podrá acceder desde la misma página web para ir viendo el avance de la impresión 3D realizada por el brazo.

2. Ingeniería de la solución

2.1 Especificaciones técnicas:

- El brazo robótico debe tener un rango de movimiento adecuado para cubrir el área de impresión mínima de 300x300x300.
- El brazo robótico debe estar equipado con una herramienta de extrusión que pueda imprimir tanto pla como abs y el mismo cabezal debe ser capaz de intercambiarse fácilmente
- Se debe operar mediante una red wifi a partir de una página web con una interfaz amigable con el usuario la cual sea capaz de cargar un archivo parametrizado directo para imprimir y poder inicializar la operación así como poder ver mediante el aplicativo el proceso en tiempo real
- La conexión inalámbrica debe ser estable y segura para garantizar una comunicación fluida entre la impresora y el aplicativo.
- La estructura del brazo robótico debe ser robusta y resistente para garantizar una operación estable y segura.
- El costo de fabricación del brazo robótico debe ser lo más bajo posible sin comprometer la calidad y la eficiencia del sistema.
- El brazo robótico debe ser fácil de desmontar y mantener para facilitar la reparación y el reemplazo de piezas en caso de fallos.Principio del formulario
- La estructura debe ser estable y de diseño simple que garantice posibles aplicaciones de dimensionamiento a futuro.

2.2 Detalle de pines a utilizar

Cant de pines	Elemento a controlar
2	Motor 1
2	Motor 2
2	Motor 3
2	Pantalla i2c
8	Total

2.3 Materiales a utilizar

2.3.1 Presupuesto

Artículo	Precio USD oficial	Precio $
Pololu x3	10,09	2250
Motores x3	-	Ya se contaba con el material
Cables x2	4,48	1000
Protoboard X2	4,48	1000
Fuente 12V	-	Ya se contaba con el material
Madera	-	Ya se contaba con el material
Varilla roscada	-	Ya se contaba con el material
Lápiz de impresión	31,39	7000
Guías	21,52	4800
Total	71,96	16050

2.3.2 Criterio de selección de cada material

ARDUINO UNO

Se seleccionó el Arduino como placa base para el movimiento de los motores debido a la gran cantidad de librerías que se pueden encontrar en el mercado para este tipo de placas, además de la facilidad que otorgan a la hora de programar la secuencia de movimiento.
No se utilizó directamente el ESP32 o el ESP32 CAM ya que, si bien puede adaptarse las librerías a este tipo de placas, el uso del wifi requería multiuso de pines para conectarse a internet y además mover los motores, lo cual generaba señales indeseadas que producían movimientos no deseados en los motores o directamente su no funcionamiento.

ESP32

Se seleccionó esta placa ya que nos ofrece la posibilidad de conectarnos a internet a través de una conexión wifi integrada sin necesidad de incorporar integrados.

ESP32 CAM

Esta placa se seleccionó debido a un criterio similar al del ESP32, nos brinda una cámara integrada capaz de transmitir video a través de wifi sin conexión física con el equipo a través de una dirección IP del dispositivo.

2.4 Especificaciones de Diseño

El mismo será construido constructivamente a partir de piezas elaboradas en impresión 3D y madera principalmente obviando los componentes electrónicos y de transmisión, el diseño será basado en la figura 1 con las modificaciones acordes a las necesidades.
Las piezas impresas en 3D fueron diseñadas por nosotros mismos. Se tuvo un enfoque de que con la menor cantidad de material posible se pueda generar una pieza con la misma

funcionalidad y que soportara las condiciones a la que se estaba sometido. Se utilizó una configuración de relleno del 20% y el material a utilizar fue PLA.
Para las piezas compuestas por madera, se utilizó una cortadora láser y tablas de [material].

[piezas impresas]

2.5 Software a utilizar

Se usará lenguaje C para programar los siguientes componentes:

- ESP32: El cual se utilizará para comunicar la página web donde se depositara el código con el ARDUINO UNO.
- ARDUINO UNO: Se utilizará esta placa de desarrollo para la transformación del código recibido por el usuario a uno que el robot pueda utilizar, y además para el comando de los motores a utilizar.
- ESP32 CAM: En este caso se utilizará desde la página web para el seguimiento de la impresión a distancia.

También se utilizará una combinación de HTML y CSS para el diseño de la página web.

Los pasos a seguir para el funcionamiento completo del proyecto son:

1. **Subida del archivo:** Se sube el archivo que se quiere imprimir a la página web. Esta realiza un HTTP POST del contenido del archivo a un archivo de Google sheets
2. **Manejo desde Google Sheets:** El archivo de Google Sheets está configurado para que cuando recibe un HTTP POST, obtenga todo el contenido que se le ha enviado y lo coloque por línea en la primera columna de la única hoja que se encuentra en el documento. De la misma forma, cuando recibe un HTTP GET, envía toda la información de la primera columna en un solo string, en el cual cada línea está separada por una coma.
3. **Obtención desde el ESP32:** El ESP32 está programado para realizar un HTTP GET al archivo del Google Sheets. Cuando obtiene el string con el código subido, ejecuta una función que elimina todas las líneas innecesarias para el funcionamiento de los motores, ya sean pre configuraciones, manejo del extrusor, etc.
4. **Envío al ARDUINO UNO:** Una vez filtrada toda la información, el ESP32 envía el nuevo string generado a través del puerto serie (TX y RX) hacia el ARDUINO UNO.
    a. Este hace un segundo manejo del string en donde, por cada línea de comando del código, obtiene las posiciones de destino tanto de X como Y.
    b. Una vez obtenidas las posiciones, manda a llamar a la función que calcula la cantidad de pasos que debe realizar cada motor según nuestro sistema de movimiento.
    c. Cuando calcula la cantidad de pasos manda a llamar a otra función que es la que comanda a los motores, diciendo que se muevan la cantidad de pasos calculados, junto con una configuración para manejar los tiempos de movimiento de cada motor.
5. **Uso del ES32 CAM:** El código del ESP32 CAM está seteado para funcionar a la inversa del ESP32, este genera una IP para la cámara que está integrada, y realiza un HTTP POST con su dirección IP a otro archivo de Google Sheets.
   Cuando la página web solicita ver el avance de la impresión, realiza primero un HTTP GET al archivo de Google Sheets donde esta la direccion IP de la cámara del

ESP32 CAM, para dirigirse a esta y mostrar el video que está filmando el ESP32 CAM con el avance de la impresión.

2.5.1 Código del ESP32

Para este caso, a fin de un mejor control, orden y revisión del código, se decidió separarlo en 2 módulos complementarios. En uno se definen las funciones que se van a utilizar mientras en el otro se utilizan y arma el código general.
El módulo general quedaría de la siguiente forma:

```
#include <HardwareSerial.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <string.h>

#define esp32TX 1  // Puedes ajustar los pines según tu configuración
#define esp32RX 3

#define WIFI_SSID "PCwifi"
#define WIFI_PASSWORD "ingmustone"
String GOOGLE_SCRIPT_ID =
"AKfycbxiWrbWpRyaDDQQgARydaQq3UQNIaPEfO08l73xLwA0IrmX1_svnYCXUWWWk37
KXSlUqg";

#define MAX_COMMAND_LENGTH 255

int SendMessage=0;
char gcode_string[1024]; // Tamaño suficiente para almacenar las líneas del archivo
int FileOk = 1;
const char* payloadCopy;
int Filas = 0;
int Movimientos[50][3]; // cantidad de filas que va contener se estima en 50 por ahora
char Mensaje[1024]="";

void setup() {
  Serial.begin(115200);
  Serial2.begin(2400, SERIAL_8N1, esp32RX, esp32TX);  // Inicializa Serial2 con pines
personalizados

  String payload;

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
```

```
    Serial.println("Conectando");
  }

  HTTPClient http;
  String url="https://script.google.com/macros/s/"+GOOGLE_SCRIPT_ID+"/exec?read";
          http.begin(url.c_str()); //Specify the URL and certificate
  http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
          int httpCode = http.GET();
    if (httpCode > 0) { //Check for the returning code
        payload = http.getString();
        payloadCopy=payload.c_str();
        Serial.println("HTTP ok");
      }
          http.end();

}

void loop() {
  if (SendMessage==0){
  strcpy(gcode_string, payloadCopy);
  Serial.println("Armando Matriz de Movimientos");
  MatrizDeMovimientos(gcode_string);
  Serial.println("Matriz Armada");

    for (int Fila=0; Fila<Filas; Fila++){
      char numero[20];
      sprintf(numero, "%d", Movimientos[Fila][0]);
      strcat(Mensaje, numero);
      strcat(Mensaje,",");
      sprintf(numero, "%d", Movimientos[Fila][1]);
      strcat(Mensaje, numero);
      strcat(Mensaje,",");
      sprintf(numero, "%d", Movimientos[Fila][2]);
      strcat(Mensaje, numero);
      strcat(Mensaje,";");
    }


    Serial2.println(Mensaje);
    SendMessage=1;
  }
}
```

Mientras que el módulo de las funciones quedaría de la siguiente forma:

```
bool TryCatch (char Letra, const char* line){
  char* Pos=0;
  //PosZ=*(strchr(line,'Z'));
```

```
  bool success = true;
  Pos=strchr(line,Letra);
  return success;
}

void extractSubstring(const char* input, char* output, size_t startIndex, size_t length) {
    size_t inputLength = strlen(input);
    if (startIndex >= inputLength) {
        // Error: startIndex está fuera de los límites de la cadena de entrada
        return;
    }
    size_t endIndex = startIndex + length;
    if (endIndex > inputLength) {
        // Ajustar endIndex si excede la longitud de la cadena de entrada
        endIndex = inputLength;
    }

    strncpy(output, input + startIndex, endIndex - startIndex);
    output[endIndex - startIndex] = '\0';
}

void MatrizDeMovimientos(char* gcode_string){
  int comandoG = 0;
  char* line;
  line = strtok(gcode_string, ",");
  while (line != NULL) {
    if (line[0] == 'G') {
      comandoG = 1;

      char newX[20]="0";
      char newY[20]="0";
      char newZ[20]="0";
      char* PosX = 0;
      char* PosY = 0;
      char* PosZ = 0;
      char* PosE = 0;
      bool Fx, Fy, Fz, Fe = true;
      size_t Px,Py,Pe,Pz=0;

      switch (line[1]){
        case '0':
          /*X e Y Siempre
          E nunca
          Z quizas*/
          //---------------------------------- Extrayendo Z si la hay      --------------------------
          if (TryCatch('Z',line)){
            PosZ=strchr(line,'Z');
            Pz = PosZ - line;
```

```
    size_t lengthZ = strlen(line) - Pz;
    extractSubstring(line, newZ, Pz + 1, lengthZ);
    Fz = true;
  } else {
    Fz = false;
  }
  //---------------------------------- Extrayendo X e Y  ---------------------------
  //obtengo la posicion de X e Y

  PosX=strchr(line,'X');
  Px= PosX - line;
  Fx = true;

  PosY=strchr(line,'Y');
  Py = PosY - line;
  Fy = true;

  if (Fx && Fy){
    size_t lengthX = Py - Px - 1;
    extractSubstring(line, newX, Px + 1, lengthX);
  }
  if (Fy && Fz){
    size_t lengthY = Pz - Py - 1;
    extractSubstring(line, newY, Py + 1, lengthY);
  } else {
    size_t lengthY = strlen(line) - Py + 1;
    extractSubstring(line, newY, Py + 1, lengthY);
  }


  Movimientos[Filas][0] = atof(newX);
  Movimientos[Filas][1] = atof(newY);
  Movimientos[Filas][2] = atof(newZ);

  break;

 case '1':
  /*X e Y quizas
  E siempre
  Z nunca*/
  //------------------ Revisando si hay X e Y --------------------------------
  if (TryCatch('X',line)){
    PosX=strchr(line,'X');
    Px= PosX - line;
    Fx=true;
  } else {
    Fx = false;
  }
```

```
        if (TryCatch('Y',line)){
          PosY=strchr(line,'Y');
          Py = PosY - line;
          Fy=true;
        } else {
          Fy = false;
        }
        if (TryCatch('E',line)){
          PosE=strchr(line,'E');
          Pe = PosE - line;
          Fe=true;
        } else {
          Fe = false;
        }
        if (Fx && Fy){
          size_t lengthX = Py - Px - 1;
          extractSubstring(line, newX, Px + 1, lengthX);
        }
        if (Fy && Fe){
          size_t lengthY = Pe - Py - 1;
          extractSubstring(line, newY, Py + 1, lengthY);
        }

        Movimientos[Filas][0] = atof(newX);
        Movimientos[Filas][1] = atof(newY);
        Movimientos[Filas][2] = 0;

        break;
      default:
        break;
    }
    Filas++;
  }
  line = strtok(NULL, ",");
 }
}
```

2.5.2 Código para el Arduino Uno

En este caso también se decidió tener dos módulos similares al caso anterior. De los cuales el código general quedaría:

```
#include <Stepper.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
```

```
String receivedMessage = "";
int MensajeRecibido = 1;

// Define el número de pasos por vuelta del motor
const int stepsPerRevolutionZ = 200; //suponiendo que es 1mm por vuelta
const int stepsPerRevolutionX = 1600; //suponiendo que la relacion es 80:1
const int stepsPerRevolutionY = 200; //la relacion es 1:1

// Define los pines de control para el driver L298N
const int motor1Pins[] = {2,3, 4, 5}; // IN1, IN2, IN3, IN4
const int motor2Pins[] = {6, 7, 8, 9};   // IN1, IN2, IN3, IN4
const int motor3Pins[] = {10, 11, 12, 13};   // IN1, IN2, IN3, IN4

int filas = 0, columnas = 3;
int Movimientos[50][3]; // cantidad de filas que va contener se estima en 50 por ahora

#define L1 21.0 // Longitud del primer eslabón en unidades de longitud
#define L2 20.0 // Longitud del segundo eslabón en unidades de longitud

#define avance_motor 1.8

// Crea objetos Stepper para cada motor
Stepper motorX(stepsPerRevolutionX, motor1Pins[0], motor1Pins[1], motor1Pins[2],
motor1Pins[3]);
Stepper motorY(stepsPerRevolutionY, motor2Pins[0], motor2Pins[1], motor2Pins[2],
motor2Pins[3]);
Stepper motorZ(stepsPerRevolutionZ, motor3Pins[0], motor3Pins[1], motor3Pins[2],
motor3Pins[3]);

int stepDelayX = 25;
int stepDelayY = 25;

float ultimo_step1 = 0;
float ultimo_step2 = 0;
float ultimo_step3 = 0;

void setup() {
  // Establece la velocidad de los motores
  motorX.setSpeed(stepDelayX); // Puedes ajustar esta velocidad según sea necesario
  motorY.setSpeed(stepDelayY);
  motorZ.setSpeed(20);
}

void loop() {

  //receivedMessage="G0 F600 X15 Y15 Z10,G1 X25 Y15 E0.94529,G1 X25 Y25
E0.94529,G1 X15 Y25 E0.94529";
```

```
  while (Serial.available() > 0 && MensajeRecibido==0) {
    receivedMessage = Serial.readStringUntil('\r');
    Serial.println("Mensaje Recibido");
    Serial.println(receivedMessage);
    //miCadena.indexOf(subCadena)
    if ( receivedMessage.indexOf(",") != -1 && receivedMessage.indexOf(";") != -1){
      MensajeRecibido=1;
    }
  }

  if (MensajeRecibido==1){
    const char* charMessage = receivedMessage.c_str();
    MatrizMovimientos(charMessage);
    for (int fila=0; fila <filas; fila++){
      motorZ.step(Movimientos[fila][2]*stepsPerRevolutionZ - ultimo_step3);
      moverRobotSCARA(Movimientos[fila][0], Movimientos[fila][1]);
      ultimo_step3 = Movimientos[fila][2]*stepsPerRevolutionZ;
    }
    Serial.println("Terminado");
    MensajeRecibido = 0;
    receivedMessage = "";  // Limpiar el mensaje recibido
    delay(1000);
  }
  for (int i = 0; i < filas; i++) {
    for (int j = 0; j < columnas; j++) {
      Movimientos[i][j] = 0;
    }
  }
  filas=0;
}
```

Mientras que el módulo de funciones quedaría de la siguiente forma:

```
void MatrizMovimientos(char texto[]){
  for (int i = 0; i < strlen(texto); i++) {
    if (texto[i] == ';') {
      filas++;
    }
  }
  char *token = strtok(texto, ",;");
  for (int i = 0; i < filas; i++) {
    for (int j = 0; j < columnas; j++) {
      Movimientos[i][j] = atoi(token);
      // Serial.print("Valor Leido: ");
      // Serial.println(Movimientos[i][j]);
      token = strtok(NULL, ",;");
    }
```

```
  }
}

void moverRobotSCARA(float x, float y) {
  // Función para mover el robot SCARA a una posición específica
  int cuadrante = 0;
  x=abs(x);
  y=abs(y);

  float d = sqrt(x * x + y * y);
  float phi = atan2(y, x);
  // Calcular los ángulos de las articulaciones mediante cinemática inversa
  float theta2 = acos((d * d - L1 * L1 - L2 * L2) / (2 * L1 * L2));
  float theta1 = phi - atan2(L2 * sin(theta2), L1 + L2 * cos(theta2));
  // Convertir los ángulos a pasos para los motores paso a paso
  float avanzar1 = ((theta1 - ultimo_step1) * 360)/(2 * PI);
  float avanzar2 = ((theta2 - ultimo_step2) * 360)/(2 * PI);

  int steps1 = avanzar1 / avance_motor;
  int steps2 = avanzar2 / avance_motor;

  MoverBrazo(steps1, steps2);

  ultimo_step1 = steps1;
  ultimo_step2 = steps2;
}

void MoverBrazo(int pasosX, int pasosY){
  int timeX = 0;
  int timeY = 0;
  int timetotal = 0;
  if (pasosX!=0 && pasosY!=0){
    float relacion = abs(pasosX)/abs(pasosY);
    stepDelayX = stepDelayX * pow(relacion,(-1));
          timetotal = abs(pasosX) * stepDelayX;
  } else if (pasosY==0 && pasosX!=0){
          timetotal = abs(pasosX)* stepDelayX;
  } else {
    timetotal = abs(pasosY)* stepDelayY;
  }

  for(int i=0;i<=timetotal;i++){
    if (i-timeX >= stepDelayX){
      if (pasosX>0){
        motorX.step(1);
      } else if (pasosX<0){
        motorX.step(-1);
      }
```

```
      timeX = i;
    }
    if (i-timeY >= stepDelayY){
      if (pasosY>0){
        motorY.step(1);
      } else if (pasosY<0){
          motorY.step(-1);
        }
      timeY = i;
           }
    delay(1);
  }
}
```

2.5.3 Código para el ESP32 CAM

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "Repetidor";
const char* password = "53EW9e527c2fAeBb3a2d";
const char* spreadsheetId = "13_MIWFoQno4I7jppms3PyK7UoSv1RYUoeWZiwnOPLt0";
const char* sheetName = "Hoja 1";
const char* apiKey = "0e1c00f2a8ca7bc3949755c1b3a4f3386d19e63f";

void setup() {
  Serial.begin(115200);
  delay(2000);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a WiFi...");
  }

  Serial.println("Conectado a WiFi");

  sendDataToGoogleSheets("Hola mundo");
}

void loop() {
  // No se requiere ninguna acción en el bucle principal
}

void sendDataToGoogleSheets(const String& data) {
  HTTPClient http;

  // Construir la URL de solicitud
```

```
  String url = "https://sheets.googleapis.com/v4/spreadsheets/";
  url += spreadsheetId;
  url += "/values/";
  url += sheetName;
  url += ":append?valueInputOption=USER_ENTERED&key=";
  url += apiKey;

  // Configurar la solicitud HTTP
  http.begin(url);
  http.addHeader("Content-Type", "application/json");

  // Construir el cuerpo de la solicitud
  String jsonBody = "{\"values\": [[\"";
  jsonBody += data;
  jsonBody += "\"]]}";

  // Enviar la solicitud HTTP POST
  int httpResponseCode = http.POST(jsonBody);

  if (httpResponseCode > 0) {
    Serial.print("Datos enviados a Google Sheets. Código de respuesta: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Error en la solicitud. Código de error: ");
    Serial.println(httpResponseCode);
  }

  http.end();
}
```

2.5.4 Código para la página web

```
<!DOCTYPE html>
<html>
<head>
    <title>Subir archivo de texto y enviar contenido</title>

    <script>
        function mostrarContenido() {
            var fileInput = document.getElementById('archivo');
            var file = fileInput.files[0];
            var reader = new FileReader();

            reader.onload = function(e) {
                var contenido = e.target.result;
                sessionStorage.setItem('contenidoArchivo', contenido);
                enviarContenido(contenido); // Llamar a la función para enviar el contenido
            };
```

```
            reader.readAsText(file);
        }

        function enviarContenido(contenido) {
            var messageElement = document.createElement('p'); // Crear un elemento <p> para
mostrar el mensaje
            document.body.appendChild(messageElement); // Agregar el elemento al cuerpo de
la página

            // Enviar contenido del archivo a través de una solicitud POST
            var url =
'https://script.google.com/macros/s/AKfycbx4OhybyTxnyrfl0gZBD7UVtPMlXKuliM4_bIY-GXV
duRwSF4Ywt0MtSQmRvjf01CrcrQ/exec'; // Reemplaza con la URL de tu hoja de cálculo
            var formData = new FormData();
            formData.append('columna1', contenido);

            fetch(url, {
                method: 'POST',
                body: formData
            })
            .then(function(response) {
                // Manejar la respuesta de la solicitud POST
                if (response.ok) {
                    messageElement.textContent = 'Contenido del archivo enviado exitosamente.';
                } else {
                    messageElement.textContent = 'Error al enviar el contenido del archivo.';
                }
            })
            .catch(function(error) {
                messageElement.textContent = 'Error en la solicitud POST: ' + error;
            });
        }

        function PedirAvance() {
            console.log("Se ha hecho clic en el botón.");
            // URL del script de Google Apps Script
            var scriptUrl =
'https://script.google.com/macros/s/AKfycbzhyA7-li2_mEbGQmF8bA8-YJE-evMOFxLcz24far
C0JJgiM2-fcK084o_NPIV4uaXSEw/exec';

            // Realiza la solicitud GET usando la función fetch
            fetch(scriptUrl)
                .then(function(response) {
                    return response.text();
                })
                .then(function(data) {
                console.log("Respuesta del servidor: " + data);
```

```
            var IP = data.replace(/[""]+/g, '');
            console.log("Respuesta del servidor: " + IP);
            window.location.href = 'http://' + IP;
            })
            .catch(function(error) {
            console.error("Error en la solicitud GET: " + error);
            });
      }
   </script>

</head>
<body>
   <h1>Subir archivo de texto</h1>
   <P>Una vez obtenido el archivo.gcode, cambie la extensión de este a .txt y súbelo</P>
   <input type="file" id="archivo" accept=".txt">
   <br><br>
   <button onclick="mostrarContenido()">Subir</button>
   <button onclick="PedirAvance()">Ver Avance</button>
   <br><br>
</body>
</html>
```

La interfaz obtenida quedaría de la siguiente manera:

2.6 Explicación del funcionamiento del código

2.6.2 Explicación código para el Arduino Uno

Vamos a detenernos específicamente en este punto para poder detallar el método de funcionamiento de la transformación de las posiciones cartesianas obtenidas a la cantidad de pasos que se debe mover cada motor junto con la velocidad a la que deben hacerlo.

```
 while (Serial.available() > 0 && MensajeRecibido==0) {
   receivedMessage = Serial.readStringUntil('\r');
   Serial.println("Mensaje Recibido");
   Serial.println(receivedMessage);
   //miCadena.indexOf(subCadena)
   if ( receivedMessage.indexOf(",") != -1 && receivedMessage.indexOf(";") != -1){
```

```
    MensajeRecibido=1;
  }
```

Este fragmento del codigo se utiliza para habilitar la lectura del puerto Serie del arduino en espera del mensaje enviado por el ESP32 con el string que posee las líneas de código a ejecutar por el robot.

```
void MatrizMovimientos(char texto[]){
 for (int i = 0; i < strlen(texto); i++) {
  if (texto[i] == ';') {
   filas++;
  }
 }
 char *token = strtok(texto, ",;");
 for (int i = 0; i < filas; i++) {
  for (int j = 0; j < columnas; j++) {
   Movimientos[i][j] = atoi(token);
   // Serial.print("Valor Leido: ");
   // Serial.println(Movimientos[i][j]);
   token = strtok(NULL, ",;");
  }
 }
```

Seguidamente la primera función a utilizar es la de la generación de la Matriz de movimientos, es decir, se arma una lista con las posiciones finales de cada eje por cada línea de código recibida. Esto me genera una lista con 3 valores en cada elemento, siendo estos valores las posiciones finales de (X, Y, Z) .

```
void moverRobotSCARA(float x, float y) {
 // Función para mover el robot SCARA a una posición específica
 int cuadrante = 0;
 x=abs(x);
 y=abs(y);

 float d = sqrt(x * x + y * y);
 float phi = atan2(y, x);
 // Calcular los ángulos de las articulaciones mediante cinemática inversa
 float theta2 = acos((d * d - L1 * L1 - L2 * L2) / (2 * L1 * L2));
 float theta1 = phi - atan2(L2 * sin(theta2), L1 + L2 * cos(theta2));
 // Convertir los ángulos a pasos para los motores paso a paso
 float avanzar1 = ((theta1 - ultimo_step1) * 360)/(2 * PI);
 float avanzar2 = ((theta2 - ultimo_step2) * 360)/(2 * PI);

 int steps1 = avanzar1 / avance_motor;
 int steps2 = avanzar2 / avance_motor;

 MoverBrazo(steps1, steps2);

 ultimo_step1 = steps1;
 ultimo_step2 = steps2;
```

}

En el caso de la función moverRobotSCARA, es la que transforma las posiciones finales en cantidad de pasos para cada motor, solamente para el caso de los motores 1 y 2 (por eso posee 2 valores de entradas), representantes de los ejes X e Y, ya que, como se definieron como ejes de tipo rotativos, es necesaria una transformación de coordenadas cartesianas a unas polares para su uso.
Debido a la configuración de ejes, se decidió utilizar el método geométrico para el cálculo de ángulos a girar en cada motor a fin de resultar más simple su cálculo. El desarrollo de este metodo lo podemos observar en las primeras lineas de la funcion; pasandolas a lenguaje matematico quedarian:

$$d = \sqrt{x^2 + y^2}$$

Calcula la distancia desde el origen al punto final donde se encuentra el extrusor

$$phi = arc\,tangente\,(y, x)$$

Para calcular el ángulo al cual se encuentra el punto final con respecto al eje horizontal

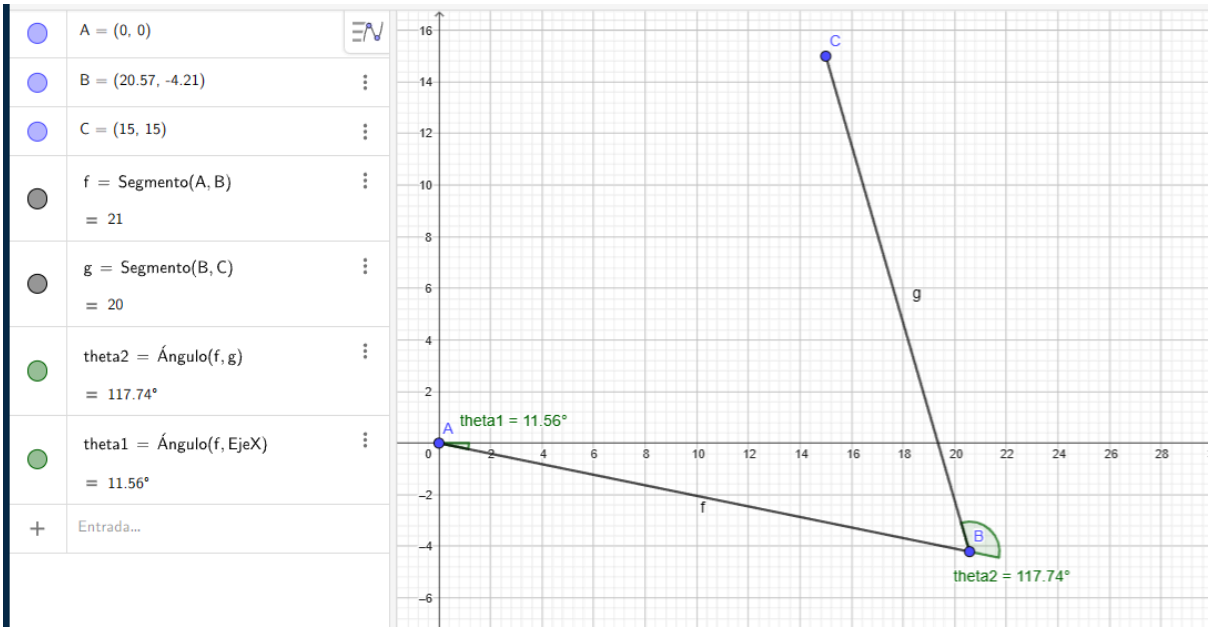
$$theta2 = arc\,coseno((d * d - L1 * L1 - L2 * L2) / (2 * L1 * L2))$$

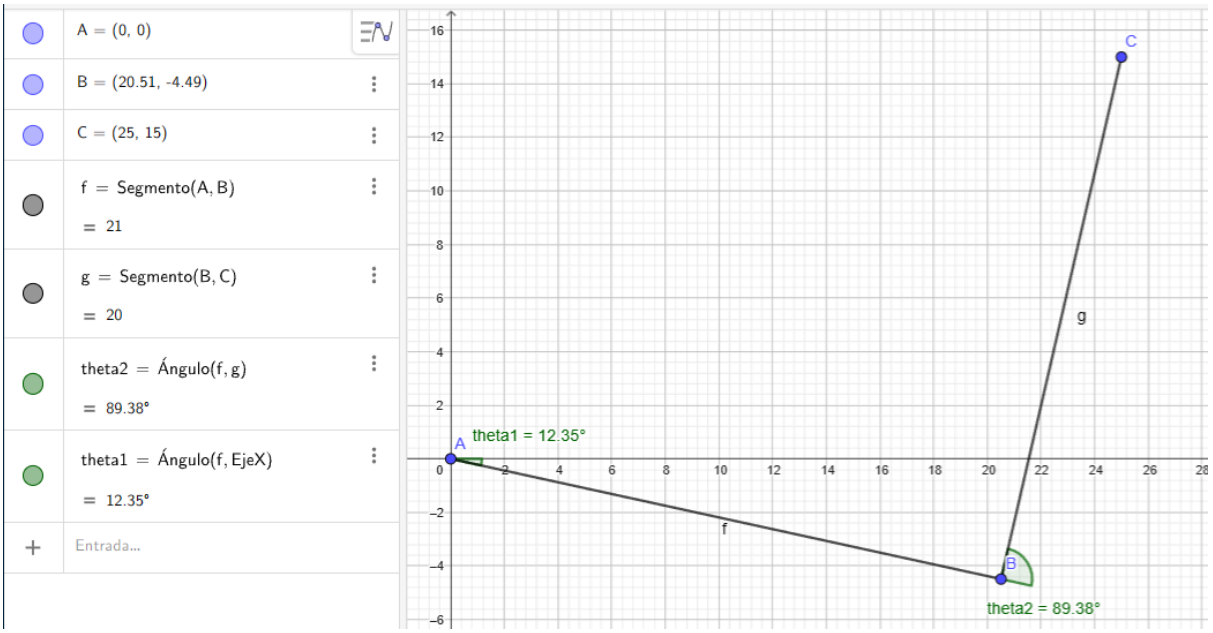
L1 y L2 representan las longitudes de los brazos 1 y 2 respectivamente.
Esta fórmula se utiliza para calcular el ángulo que hay entre el brazo 2 y la continuación de la línea central del brazo 1.

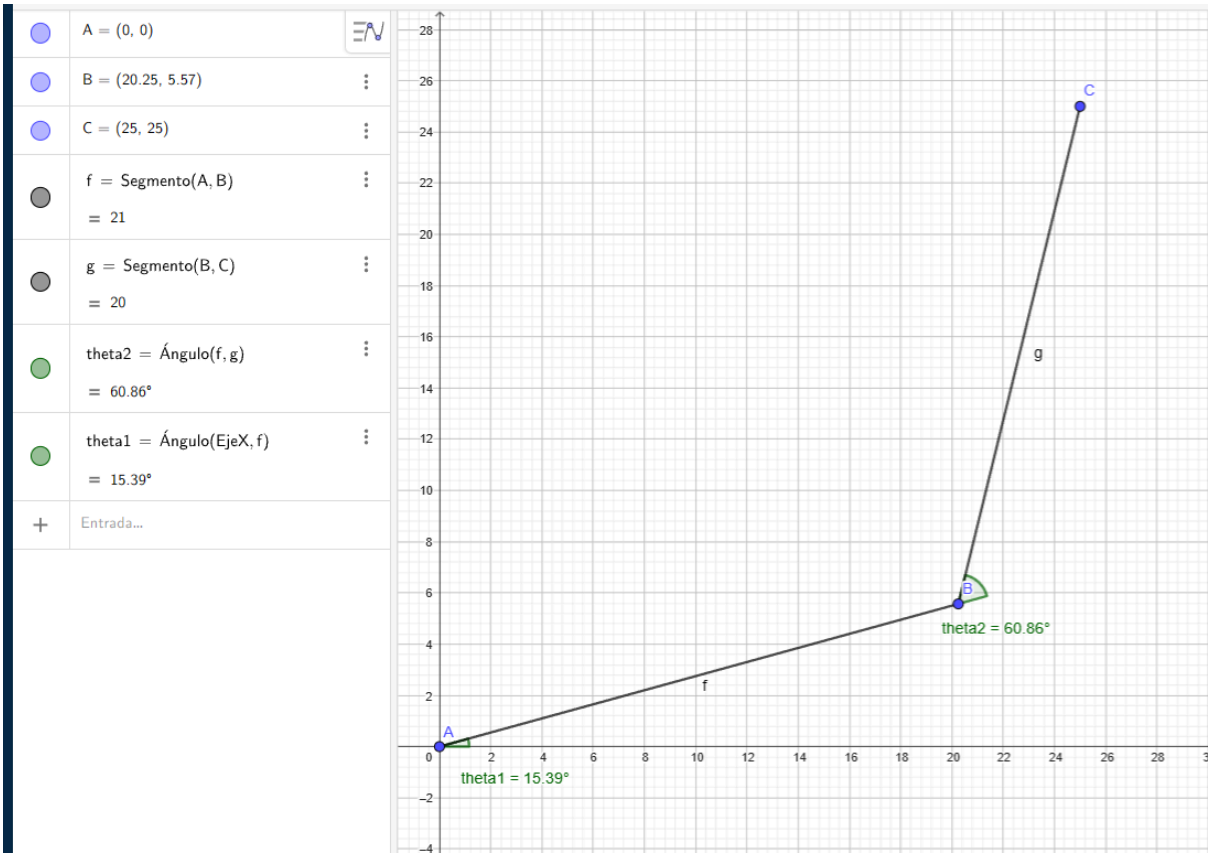
$$theta1 = phi - arc\,tangente(L2 * seno(theta2),\ L1 + L2 * coseno(theta2))$$

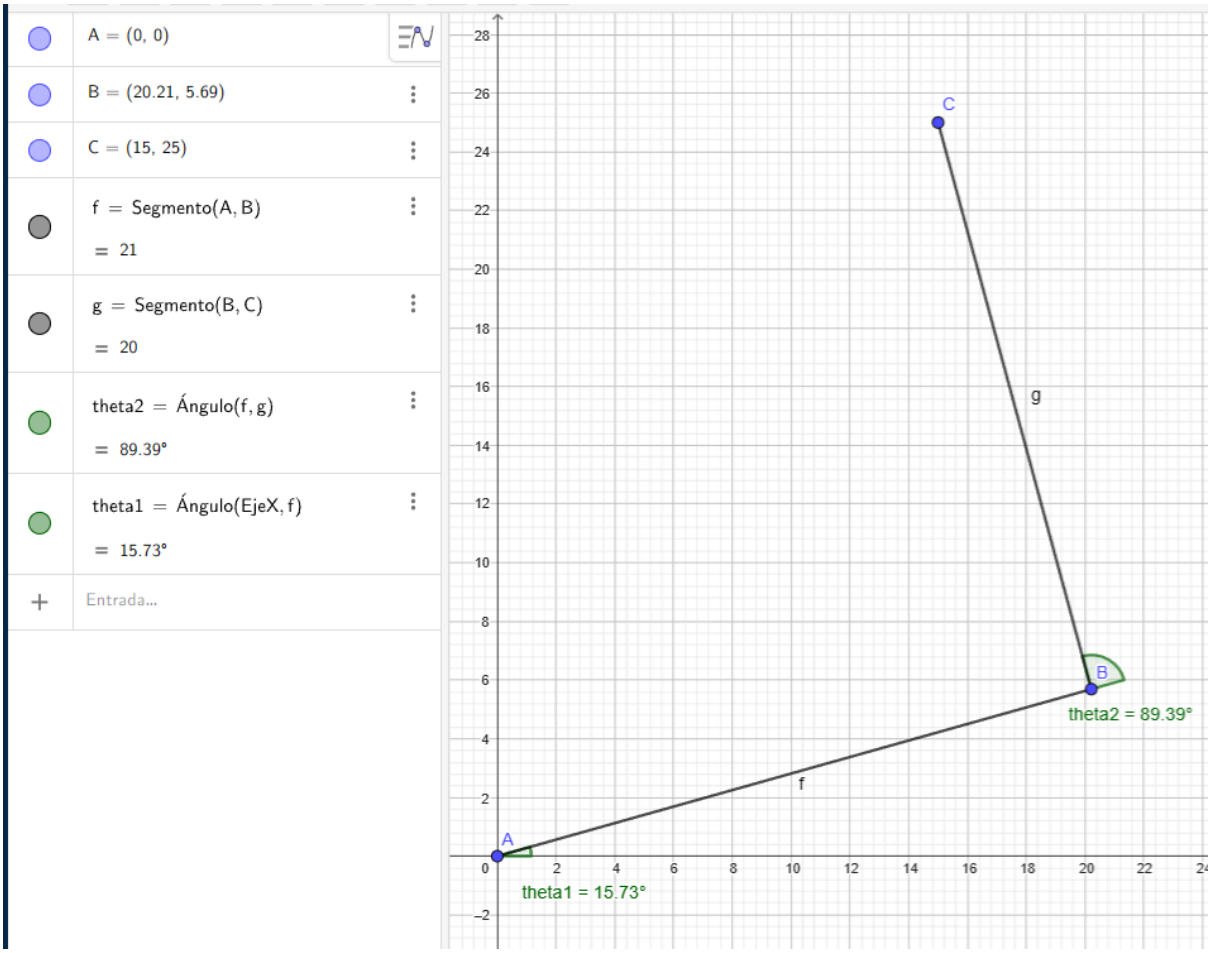
Finalmente se calcula el ángulo del brazo 1 con respecto del eje horizontal.

Con la secuencia de puntos finales (15,15); (25,15); (25,25); (15,25) los ángulos de theta1 y theta2 deberían quedar de la siguiente manera:


A = (0, 0)
B = (20.57, -4.21)
C = (15, 15)
f = Segmento(A, B)
= 21
g = Segmento(B, C)
= 20
theta2 = Ángulo(f, g)
= 117.74°
theta1 = Ángulo(f, EjeX)
= 11.56°
Entrada...
C
g
theta1 = 11.56°
A
f
B
theta2 = 117.74°


A = (0, 0)
B = (20.51, -4.49)
C = (25, 15)
f = Segmento(A, B)
= 21
g = Segmento(B, C)
= 20
theta2 = Ángulo(f, g)
= 89.38°
theta1 = Ángulo(f, EjeX)
= 12.35°
Entrada...
C
g
theta1 = 12.35°
A
f
B
theta2 = 89.38°


A = (0, 0)
B = (20.25, 5.57)
C = (25, 25)
f = Segmento(A, B)
= 21
g = Segmento(B, C)
= 20
theta2 = Ángulo(f, g)
= 60.86°
theta1 = Ángulo(EjeX, f)
= 15.39°
Entrada...
C
g
B
theta2 = 60.86°
f
A
theta1 = 15.39°


A = (0, 0)
B = (20.21, 5.69)
C = (15, 25)
f = Segmento(A, B)
= 21
g = Segmento(B, C)
= 20
theta2 = Ángulo(f, g)
= 89.39°
theta1 = Ángulo(EjeX, f)
= 15.73°
Entrada...
C
g
B
theta2 = 89.39°
f
A
theta1 = 15.73°

```
float avanzar1 = ((theta1 - ultimo_step1) * 360)/(2 * PI);
float avanzar2 = ((theta2 - ultimo_step2) * 360)/(2 * PI);

int steps1 = avanzar1 / avance_motor;
int steps2 = avanzar2 / avance_motor;
```

A continuación se calcula los ángulos que deben avanzar (o retroceder), cada motor, esto se calcula convirtiendo los ángulos calculados a grados geométricos, ya que inicialmente se encuentran en valores no representativos, es decir, valores no utilizables como grados geométricos.
También se le restara el angulo en el que el motor se encuentra actualmente, reflejado en las variables ultimo_step1 y ultimo_step2, las cuales, luego de cada movimiento se van a ir actualizando. Por ejemplo, si el motor X tiene que posicionarse en 15,73° y actualmente se encuentra en los 15,39°, solamente debería girar 0,34°, que es el valor que se guarda en la variable avanzar1 (la variable avanzar2 es homóloga a avanzar1 pero para el motor 2)
Finalmente, a esa cantidad de grados a moverse en cada motor, se lo divide por los grados por pasos del motor (en este caso seria 1.8 grados por paso, que está almacenado en la variable avance_motor), obteniendo la cantidad de pasos a realizar para avanzar esa cantidad de grados geométricos.

```
MoverBrazo(steps1, steps2);
```

Una vez terminada toda la serie de cálculos se manda a llamar la funcion MoverBrazo, que es la encargada de coordinar los movimientos de los motores 1 y 2 para que lleguen a su destino al mismo tiempo, generando una línea recta como trayectoria.

```
void MoverBrazo(int pasosX, int pasosY){
  int timeX = 0;
  int timeY = 0;
  int timetotal = 0;
  if (pasosX!=0 && pasosY!=0){
    float relacion = abs(pasosX)/abs(pasosY);
    stepDelayX = stepDelayX * pow(relacion,(-1));
          timetotal = abs(pasosX) * stepDelayX;
  } else if (pasosY==0 && pasosX!=0){
          timetotal = abs(pasosX)* stepDelayX;
  } else {
    timetotal = abs(pasosY)* stepDelayY;
  }

  for(int i=0;i<=timetotal;i++){
    if (i-timeX >= stepDelayX){
      if (pasosX>0){
        motorX.step(1);
      } else if (pasosX<0){
        motorX.step(-1);
      }
      timeX = i;
    }
```

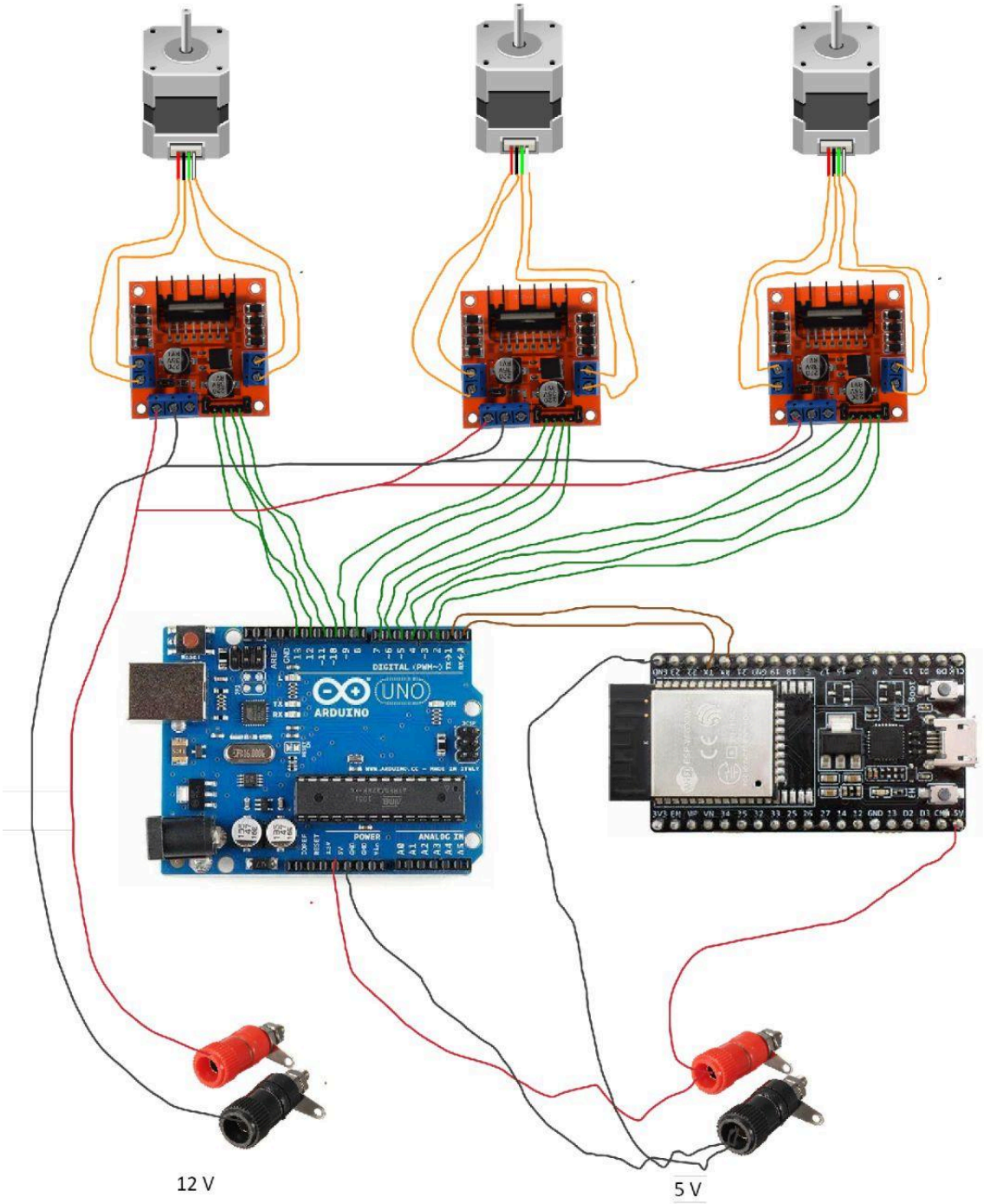
```
    if (i-timeY >= stepDelayY){
      if (pasosY>0){
        motorY.step(1);
      } else if (pasosY<0){
          motorY.step(-1);
        }
      timeY = i;
           }
    delay(1);
  }
}
```

Para que manos motores lleguen al punto de destino al mismo tiempo, sin importar la cantidad de pasos que deba dar cada unos, había varias posibilidades, como por ejemplo utilizar la librería milis de arduino, pero en este caso se decidió hacer una función con un funcionamiento similar pero con delay muy cortos.
El funcionamiento de la función MoverRoborSCARA realiza el cálculo de la relación de pasos que debe dar el motor 1 y el 2, una vez obtenido, afecta al delay o velocidad interna del motor con menos pasos por ese factor, y se obtiene que ambos motores tarden lo mismo aunque den diferente cantidad de pasos. Cabe aclarar que ambos motores ya tienen una velocidad mínima para el movimiento, por eso el factor siempre es aditivo.
Una vez calculado el tiempo que ambos motores tardaran en completar sus correspondientes pasos, se crea un bucle for que itere por cada milisegundo del total calculado, y al final se le coloca un delay de 1 milisegundo, esto hacer que cada ciclo del bucle se ejecute cada 1 milisegundo. Esto nos permite ver si, el delay de cada motor, es decir, la cantidad de milisegundos que tardan en dar cada paso, corresponde con el número de ciclo que se está ejecutando, el motor se mueva un paso. Esto afecta a ambos motores al mismo tiempo.
Cuando se tiene que mover un paso cualquiera sea el motor, se agrega un condicional para saber si se tiene que mover en sentido horario o antihorario, también podría decirse, si se tiene que mover un paso o menos un paso.
Como se puede ver es muy similar el funcionamiento de la librería milis de arduino pero con un agregado de mayor control sobre el tiempo de duración total.

2.7 Diagramas de conexionado

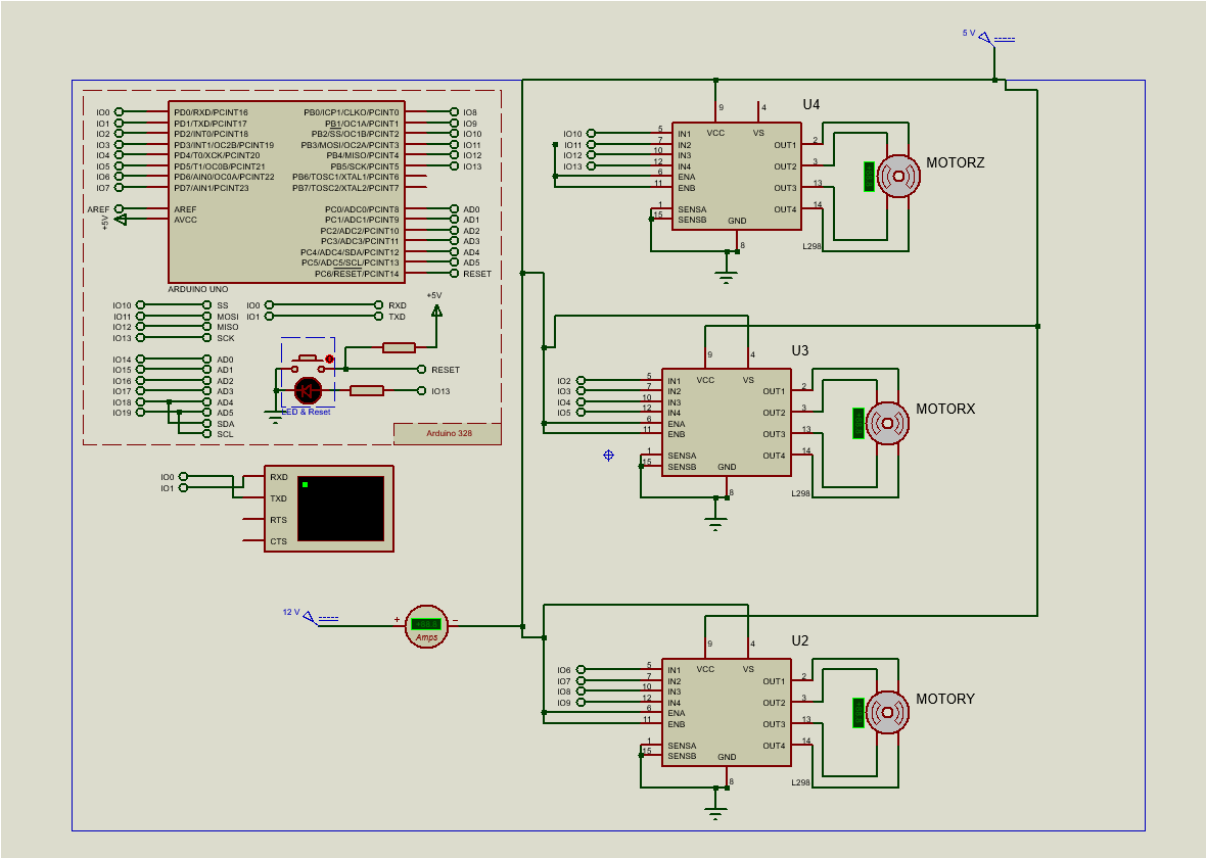
2.7.1 Conexión ARDUINO - ESP32

La conexión entre todos los componentes quedaría de la siguiente manera.



2.7.2 Simulación en Proteus

A fin de mejorar las pruebas y tener una visión más exacta sobre el movimiento de los motores, se decidió simular el circuito y el código del Arduino Uno en el software de simulación electrónica Proteus. El conexionado nos quedaría de la siguiente manera:


5 V
U4
MOTORZ
L298
U3
MOTORX
L298
U2
MOTORY
L298
ARDUINO UNO
Arduino 328
LED & Reset
RESET
12 V
Amps

3. Conclusión

Para finalizar, se logró dimensionar, fabricar y utilizar un un brazo robótico que funciona como impresora 3D. Utilizando 3 placas de desarrollo como lo son Arduino Uno, ESP32 y ESP32 CAM.
El diseño fue hecho con una eje de movimiento prismático para el movimiento vertical (en Z) del brazo, y otros 2 rotativos que se utilizan para el en X e Y, con una distribución tipo SCARA.