Instituto Politécnico Nacional

Escuela Superior de Ingeniería Mecánica y Eléctrica

Unidad Zacatenco

Materia: **Microprocesadores.**

Proyecto Final: **Brazo Robot.**

Fecha de:

- **Inicio del proyecto: 30/10/2023**
- **Finalización del proyecto: 17/01/2024**
- **Entrega del reporte: 18/01/2024**

Profesor: **Galicia Galicia Roberto.**

Grupo: **6CM7**

Alumno:

- **Córdoba Castro Isaías……. 2020303030**

Introducción:

En el siguiente reporte se hablará de los principales pasos a seguir para la elaboración de un Brazo Robótico, con finalidad aplicar los conocimientos adquiridos durante el semestre en curso, dicho proyecto está pensando para operar con uno o más microcontroladores. Para este proyecto se utilizaron 2 (Raspberry Pi 4 y Arduino nano)

Objetivo general:

Aprender el uso básico de los microcontroladores actuales como la Raspberry Pi 4, en conjunto con los microcontroladores más arcaicos. Todo con el fin unir los conocimientos obtenidos desde los cursos básicos de los semestres anteriores hasta el actual, para la elaboración de un proyecto final.

Objetivo específico:

Seleccionar y realizar un proyecto utilizando la Raspberry Pi 4 (y de ser necesario otro microcontrolador) que permita aplicar los conceptos y conocimiento de las practicas del curso (Servidores, Botones, RGB etc.), así como la importancia de saber utilizar de forma correcta los microcontroladores modernos en la vida real.

Marco teórico:

A lo largo del semestre, nuestro profesor nos hizo énfasis de lo que llamamos “Cliente-Servidor” lo cual podemos destacar lo siguiente:

En términos generales, la relación cliente-servidor se refiere a la interacción entre dos programas o dispositivos que trabajan conjuntamente para realizar tareas específicas. Esta relación es fundamental en arquitecturas de red y sistemas distribuidos. Aquí hay algunos conceptos clave:

1. **Cliente**
- Un cliente es un dispositivo o programa que solicita servicios o recursos a otro programa o dispositivo, conocido como servidor.
- En el contexto de la web, un navegador web (como Chrome, Firefox o Safari) es un ejemplo de cliente. También hay clientes de correo electrónico, clientes FTP, etc.
- Los clientes envían solicitudes al servidor para obtener información o realizar acciones.

2. **Servidor:**
- Un servidor es un dispositivo o programa que proporciona servicios, recursos o información a los clientes.
- Los servidores están siempre activos y escuchan las solicitudes de los clientes, respondiendo de manera adecuada a esas solicitudes.
- Ejemplos de servidores incluyen servidores web (como Apache o Nginx), servidores de correo, servidores de bases de datos, etc.

3. **Comunicación Cliente-Servidor:**

- La comunicación entre el cliente y el servidor generalmente sigue el modelo de solicitud y respuesta.
- El cliente envía una solicitud al servidor, que procesa la solicitud y envía una respuesta de vuelta al cliente.
- Los protocolos comunes para esta comunicación incluyen HTTP/HTTPS para la web, SMTP/IMAP para el correo electrónico, FTP para la transferencia de archivos, entre otros.

4. Arquitecturas Cliente-Servidor:

- Hay varias arquitecturas cliente-servidor, como la arquitectura de dos capas, tres capas y n capas, que definen cómo se distribuyen las funciones y responsabilidades entre el cliente y el servidor.

En resumen, la relación cliente-servidor es esencial en la informática moderna y facilita la distribución de tareas y recursos en una red. Los clientes solicitan servicios, y los servidores responden proporcionando esos servicios o recursos solicitados.

Desarrollo (Resumido):

Paso 1: Elegir proyecto, en este caso se opto por un brazo robot.

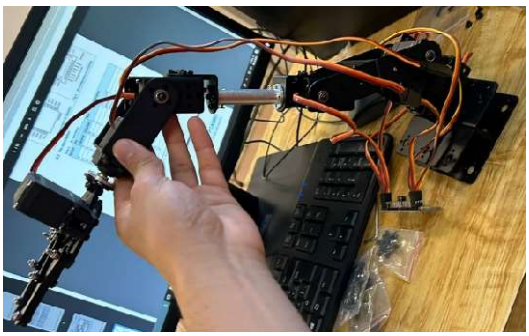
Paso 2: Obtener todos los componentes a utilizar

Materiales:

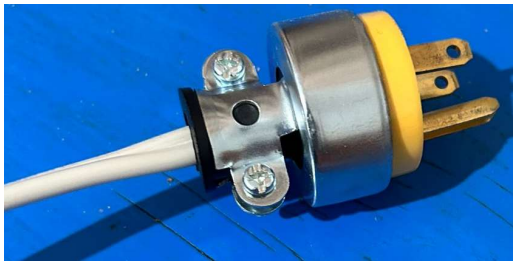
- Raspberry Pi 4.
- Arduino Nano.
- Cable dúplex.
- Fuente de 5 Volts a 10 Amperios.
- Brazo robot.
- Servos MG996R.
- Protoboard.
- Bush Botón.
- Joysticks (3).
- Jumpers o Cable calibre 22.
- Led
- Resistencia de 330 Ohm.
- Clavija
- PCA

Paso 3: Armar ciertas partes del robot (No puede armarse todo tan rápido debido a que se deben ajustar los servos).

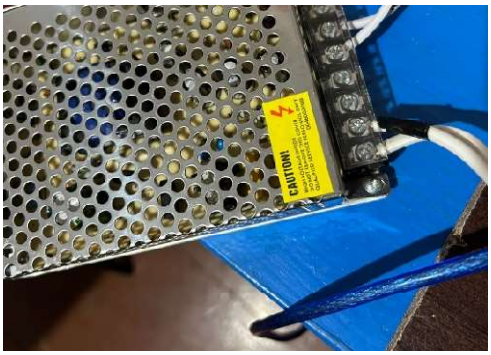
Paso 4: Calibrar los servos para tener una posición inicial.



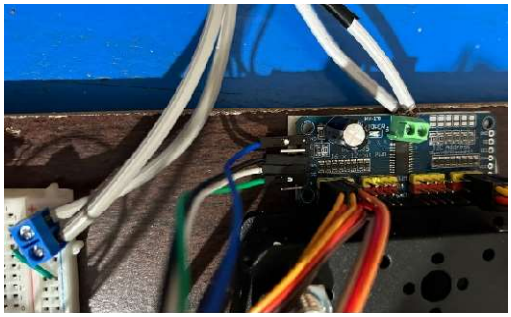
Paso 5: Ensamblar el cable dúplex a la clavija.



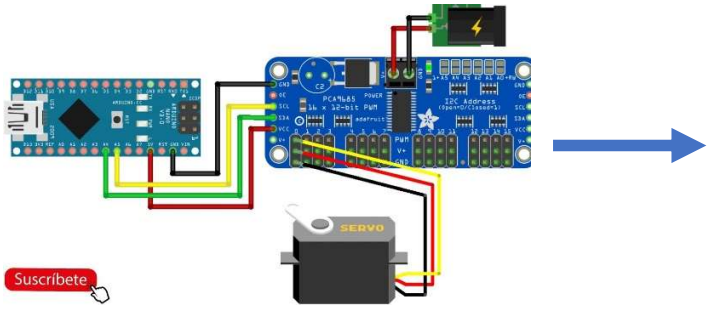
Paso 6: Unir el otro extremo del cable a la fuente para que obtenga energía.

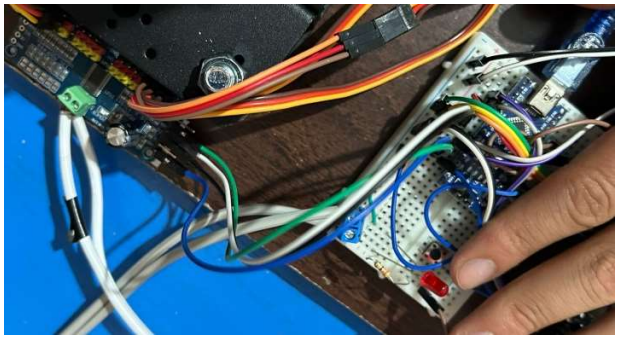


Paso 7: Realizar una conexión en paralelo a la fuente para alimentar con 5 V la Protoboard y la PCA.

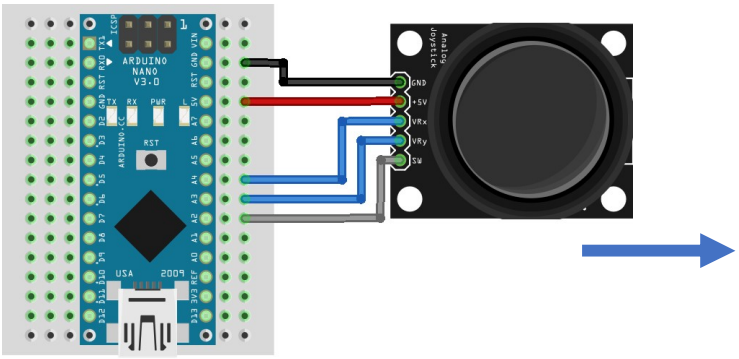


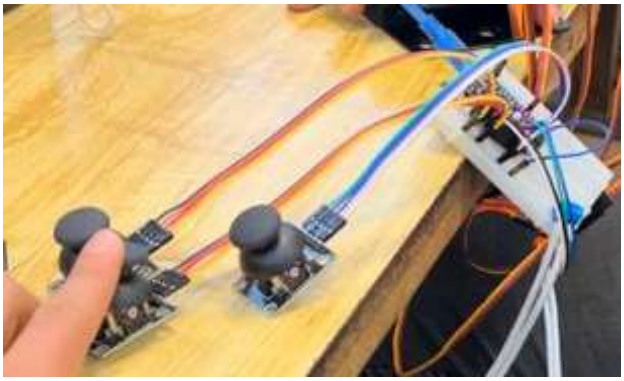
Paso 8: Conectar la PCA al Arduino NANO.



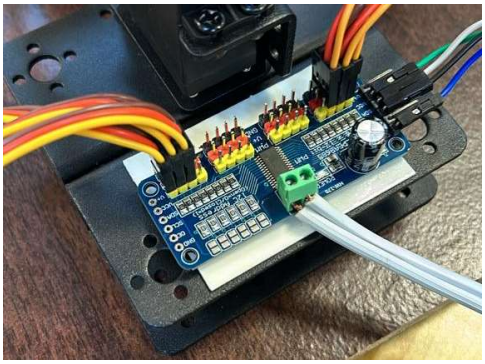


Paso 9: Conectar los Joystick al Arduino nano.

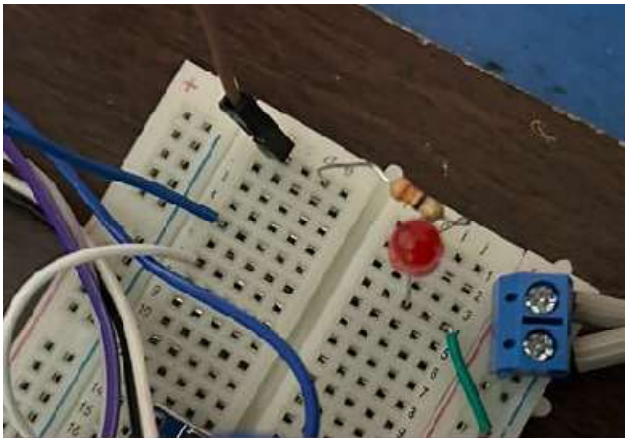




Paso 10: Conectar los servos a la PCA



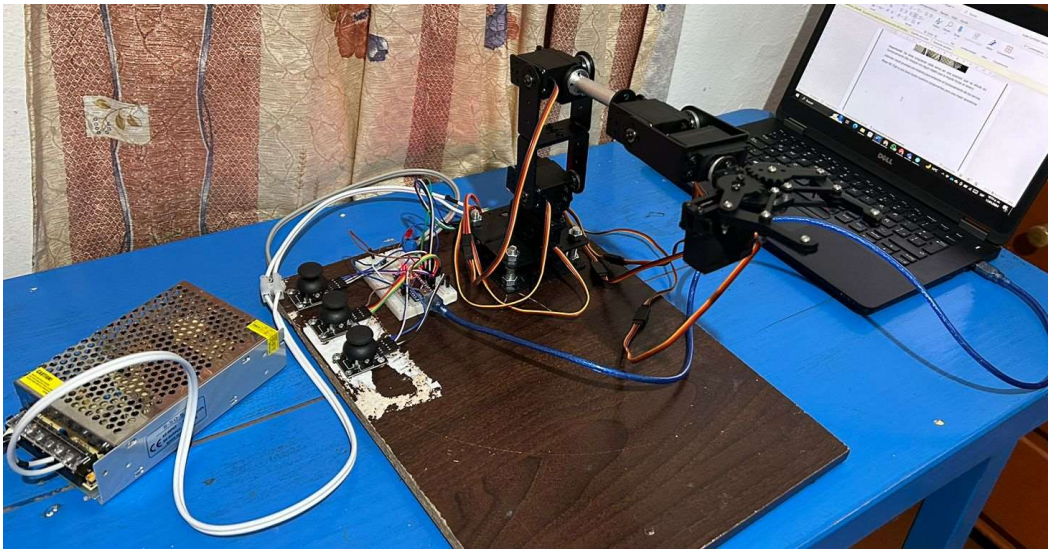
Paso 11: Crear un circuito que nos permita saber cuándo el robot esta en funcionamiento con servidor y cuando con Joystick.



Importante: Se debe programar cada servo en una posición que no afecte su funcionamiento (No choque con algún objeto que le impida forzar al motor).

Además, hacer pruebas con motores para entender el funcionamiento de los servos.

Paso 12: Fijar a una base todos nuestros componentes para una mejor apariencia.



Código:

```
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

#include <ArduinoJson.h>

#ifdef __AVR__

#include <avr/power.h>

#endif
```

Estas líneas son comunes en proyectos de Arduino que involucran el control de servomotores y la manipulación de datos en formato JSON. Son librerías que nos ayudan a tener los paquetes necesarios durante el trabajo

```
StaticJsonDocument<200> doc;
int pos0 = 102;
int pos180 = 512;
int JoyX = A0;
int JoyY = A1;
int JoyX1 = A2;
int bot2 = 2;
int bot3 = 3;
int Led2=10;
```

En esta Estas líneas de código establecen algunas variables que parecen estar relacionadas con el control de servomotores y la lectura de valores analógicos desde un joystick. La clase StaticJsonDocument se utiliza para manejar documentos JSON, pero en estas líneas de código no se está utilizando directamente, por lo que su propósito específico dependerá de otras partes del código.

```
void setup() {
 Serial.begin(9600);
  pwm.begin();
  pwm.setPWMFreq(50);
  while (!Serial) continue;
  pinMode(Led2, OUTPUT);
  initial_position();
  delay(500);
  pinMode(bot2, INPUT_PULLUP);
  pinMode(bot3, INPUT_PULLUP);
  pinMode(bot6, INPUT_PULLUP);
  pinMode(bot7, INPUT_PULLUP);
}
```

Esta sección del código de configuración (setup()) establece la comunicación serie, configura el controlador de PWM, inicializa pines, y realiza algunas configuraciones iniciales para el sistema. La función initial_position() es importante para la inicialización del sistema

```
void loop() {
 if(digitalRead(bot7)==0){
    da=1;
 Serial.println(1); }
  if(digitalRead(bot6)==0){
    da=2;
    Serial.println(2);
  }
  if(digitalRead(bot3)==0){
    da=3;
    Serial.println(2);
  }
```

Esta sección verifica el estado de tres botones (bot7, bot6, y bot3) y asigna valores a la variable da dependiendo de cuál de los botones esté presionado. Además, imprime el número correspondiente en el puerto serie.

```
switch(da){
  case 1:
    digitalWrite(Led2, HIGH);
    local();
    break;
  case 2:
    digitalWrite(Led2, LOW);
    cliente();
    break;
  case 3:
    digitalWrite(Led2, LOW);
    rutina();
    break;
  }
```

Este bloque de código controla el comportamiento del sistema según el valor de la variable da. Dependiendo de si da es 1, 2 o 3, enciende o apaga un LED (Led2) y llama a funciones específicas (local(), cliente(), rutina()).

```
void cliente(){
  if (Serial.available()) {
  String data = Serial.readStringUntil('\n');
  Serial.println(data);
  DeserializationError error = deserializeJson(doc, data);
  // Test if parsing succeeds.cd ..
  if (error) {
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    return;
  }
  }
```

Este código es una parte típica de un programa Arduino que espera recibir datos JSON por el puerto serie, los imprime para depuración y luego intenta deserializarlos utilizando ArduinoJson.

```
void local(){
 if (!digitalRead(bot2))
 {
  Servo0Position=90;
  Servo1Position=80;
  Servo2Position=10;
  Servo3Position=90;
  Servo4Position=90;
  Servo5Position=0;
  setServo(0, Servo0Position);
  setServo(1, Servo1Position);
  setServo(2, Servo2Position);
  setServo(13, Servo3Position);
  setServo(14, Servo4Position);
  setServo(15, Servo5Position);
 }
```

Esta función local() configura posiciones específicas para varios servomotores cuando el botón conectado al pin bot2 no está presionado. Debemos asegurarnos de tener la implementación adecuada de la función setServo y de haber configurado correctamente los canales de los servomotores según el hardware

```
if (analogRead(JoyX) > 800)
 {
  if (Servo0Position < 400)
  {
   Servo0Position++;
  }
 }
 if (analogRead(JoyX) < 300)
 {
  if (Servo0Position > 10)
  {
   Servo0Position--;
  }
 }
 delay(2);
 setServo(0, Servo0Position);
```

Este bloque de código ajusta la posición de un servomotor en función de la lectura analógica del joystick conectado al pin JoyX. El servomotor se mueve hacia la derecha si el joystick está inclinado hacia la derecha y hacia la izquierda si el joystick está inclinado hacia la izquierda. La velocidad de movimiento del servomotor está controlada por la velocidad con la que se mueve el joystick.

Importate realizar este procedimiento para cada servo

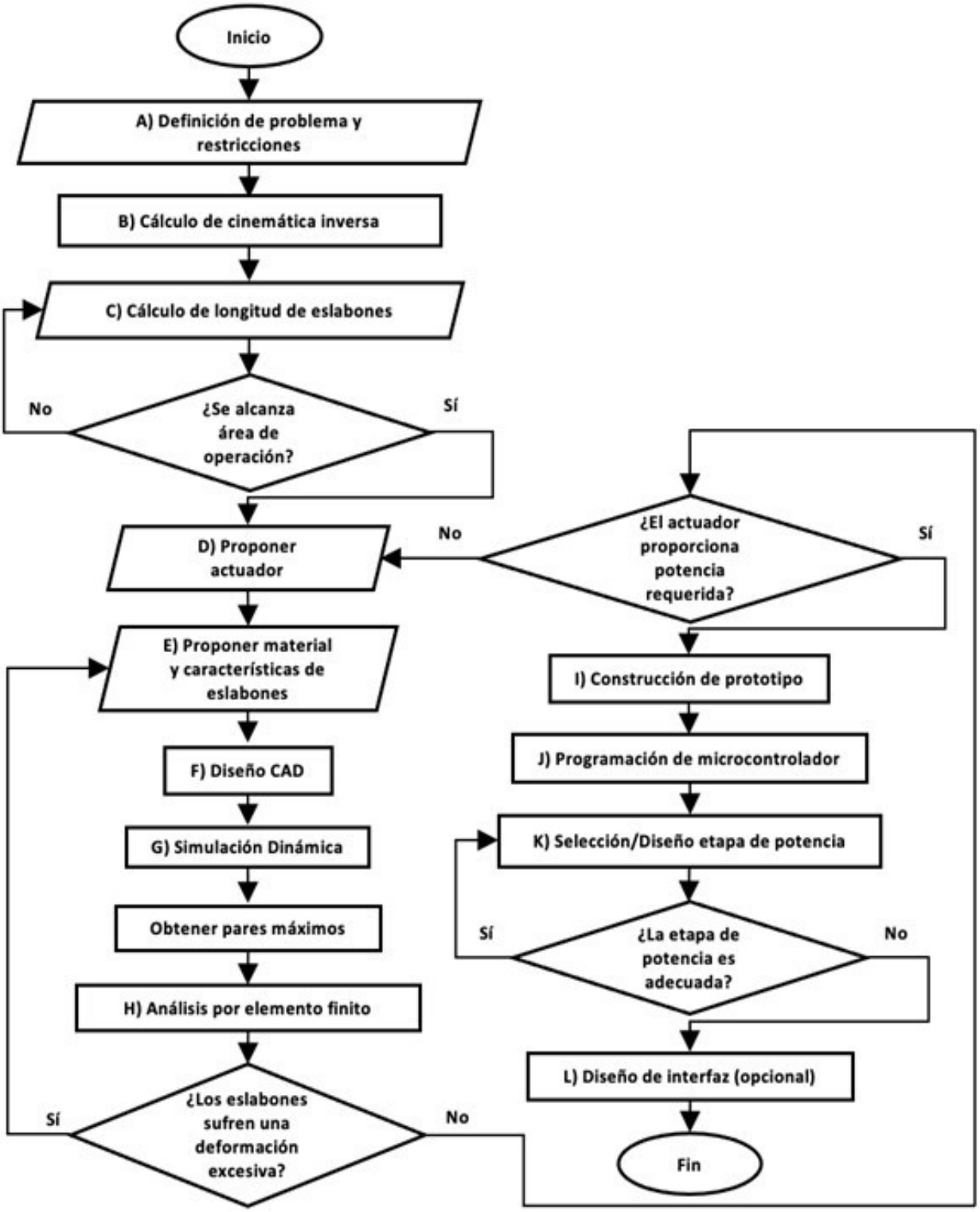
```
void rutina() {
  delay(1000);
  setServo(0, 90);
  setServo(1, 80);
  setServo(2, 10);
  setServo(13, 90);
  setServo(14, 90);
  setServo(15, 0);
  delay(1000);
```

La función rutina() ejecuta una secuencia predefinida de posiciones para varios servomotores con intervalos de retardo específicos entre cada cambio de posición. Con el fin de crear una rutina en la que el usuario tenga la única participación de oprimir un botón

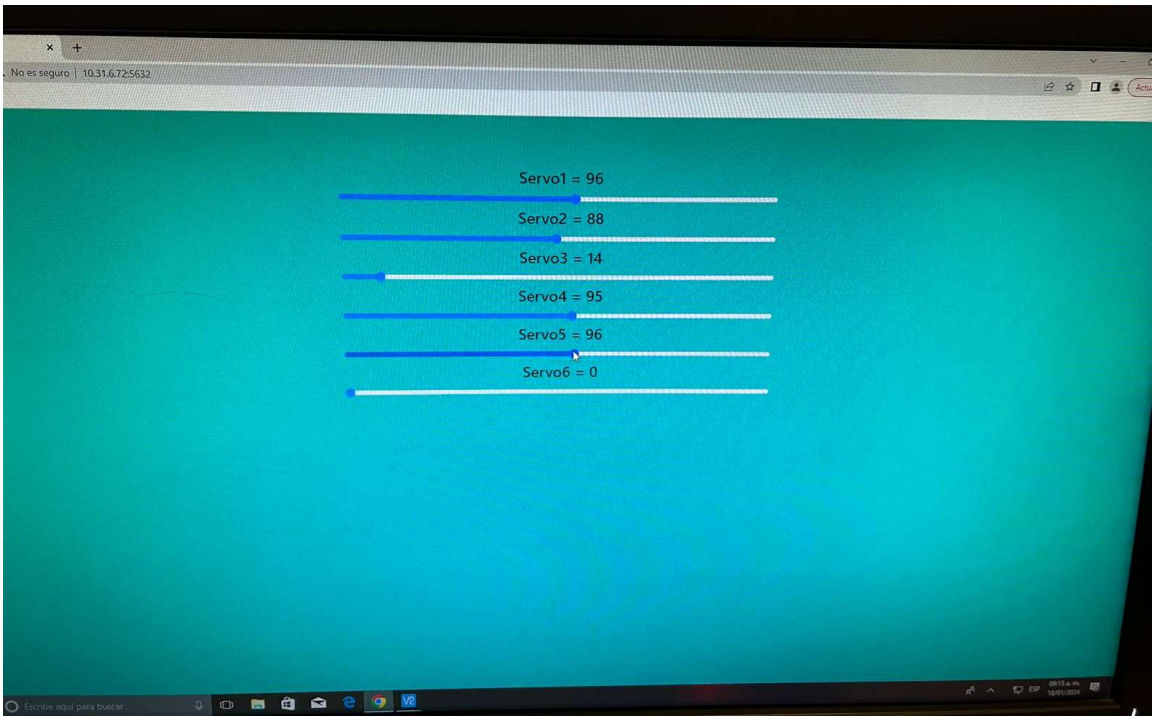
```
void setServo(int servo, int angle) {
  int duty;
  duty = map(angle, 0, 180, pos0, pos180);
  pwm.setPWM(servo, 0, duty);
}
```

La función setServo realiza la conversión del ángulo deseado a un valor de ciclo de trabajo y utiliza el controlador PWM para configurar la posición de un servomotor en el canal especificado. Este tipo de función es común en programas de Arduino para controlar servomotores.

Diagrama de flujo



Página Web



Conclusión

Este Proyecto fue bastante satisfactorio en cuestión de esfuerzo, puesto que pude apreciar el comportamiento de la teoría de circuitos que he aprendido a lo largo de la carrera, pero de una forma real, como energizar varios Servos al mismo tiempo con placa PCA, cuya alimentación provenía de una fuente con características especificas para mi circuito, además de cosas mas comunes como pelar, taladrar, conectar, ajustar y armar todo lo necesario para el funcionamiento del proyecto.

Por otra parte, aprendí a aprovechar las nuevas tecnologías utilizadas en mi carrera y en la época en la que vivo, como las IA con el fin de apoyarme a resolver problemas con una mayor eficiencia, además de poder aprender sobre los microcontroladores más modernos y su eficiencia al resolver problemáticas con una mayor eficiencia.