UNIVERSIDAD AUTONOMA DEL ESTADO DE MEXICO

FACULTAD DE INGENIERIA

UNIDAD DE APRENDIZAJE:

ROBOTICA

INTEGRANTES:

LUIS ANGEL GARCIA FEREGRINO

DAISY MONTSERRAT SANCHEZ HERNANDEZ

JESUS ALEJANDRO FELIX GONZALEZ

ULISES BECERRIL VALDES

PROFESOR: JAIME GARCIA GARCIA

# TAREA 4. IMPLEMENTACIÓN DE CINEMÁTICA INVERSA

**Objetivo.**

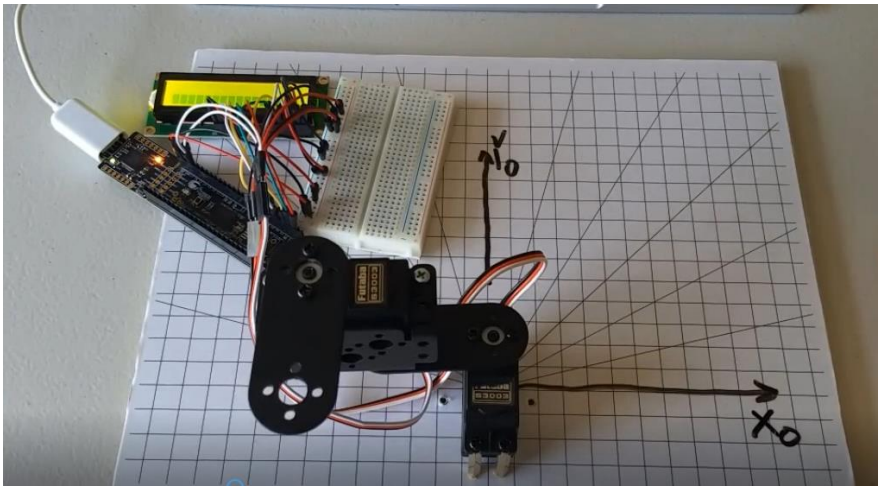
El alumno comprobara el funcionamiento de la Implementación de Cinemática Inversa a través de esta práctica.

## Introducción.

La cinemática inversa es una rama de la robótica y la ingeniería que se encarga de determinar la posición y orientación final de un robot o sistema mecánico en función de las coordenadas de sus articulaciones o componentes individuales. Es un concepto fundamental para el control y el diseño de robots, ya que permite calcular la posiciona y orientación de un robot en el espacio en respuesta a los movimientos de sus articulaciones.

## Investigación previa.

1.- Realizar la programación e implementación de un robot planar RR o de dos gados de libertad controlado por servos de entretenimiento (que pueden controlarse en un ángulo comprendido de 0 a 180°), los cuales deben estar colocados en una maqueta que tenga indicado en centímetros un cuadro cartesiano para ubicar la posición del elemento final.




El robot debe ser controlado por un microcontrolador puede ser Arduino . El programa debe tener la opción, para mover el robot utilizando la opción de cinemática inversa, el menú de usuario debe solicitar la dirección x,y.
Al iniciar la secuencia o el programa, el robot debe estar en una posición de inicio.

2. Si se basa en el programa que se encuentra en el siguiente enlace:
https://github.com/AymenNacer/Forward-and-Inverse-Kinematics-for-2-DOF-Robotic-arm
el cual esta modificado en el siguiente enlace:
https://wokwi.com/projects/378803068101648385
Se solicita que se realice la programación para calcular el ángulo θ2 por la ecuación

$$q_2 = \pm 2\ \text{atan}(\text{sqrt}((\ (l_1 + l_2)^2 - (X^2 + Y^2)\ )/\ ((X^2 + Y^2) - (l_1 - l_2)^2\ )))$$

Mostrar en pantalla las dos soluciones y seleccionar la que se apague al funcionamiento correcto del servo

3. Mejorar el programa para validar datos proporcionados por el usuario y limitar condiciones

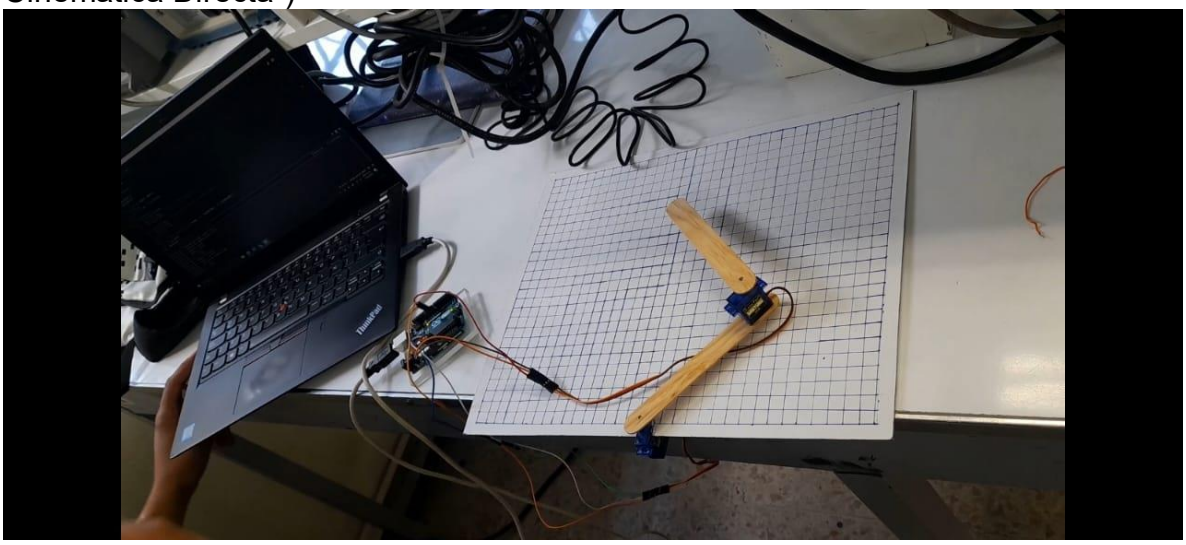
Se debe entregar un reporte, el cual debe contener:

1. Portada
2. Reporte, en donde venga el código del programa
3. Capturas de pantalla, que incluya la simulación con wokwi
4. Video o fotos del funcionamiento
5. Comparación de resultados con roboanalyzer
6. Referencias
7.

Primeramente, se debe de tener en consideración que los materiales y herramientas a utilizar son las siguientes:

Arduino IDE
Arduino o ESP32
Jumpers
Protoboard
2 o 3 servomotores (SG90 o MG995)
Una caja o cartón para realizar el cuadro cartesiano

Iniciando con lo más “fácil”, para el cuadro cartesiano utilizamos un cartón en el que pusimos las medidas correspondientes en centímetros dejando el espacio adecuado del servomotor base, el cual fue el MG995, por lo que las medidas son mostradas a partir de 2 cm, y queda de la siguiente manera después de marcar los grados de 0 a 180° de 15 en 15°. (El mismo que se usó en la Implementación de Cinemática Directa”)



Una vez teniendo lo anterior, se desarrolla la tarea. Y para el programa/código a utilizar, se hizo uso del programa proporcionado por el docente, pero, se le hicieron algunos cambios para que en vez de manipular dos servomotores, se puedan controlar 3, así mismos otra de las modificaciones que se realizaron en el código

fue que, si el usuario ingresa un ángulo fuera del rango de 0 a 180°, este se marque como incorrecto, y se le pide al usuario volverlo a ingresar, los pines que se ocupan en el Arduino son el 9, 10 y 11, se seleccionaron estos porque son PWM. Así mismo, se debe de tener en cuenta la hoja de datos de cada servomotor a utilizar, las cuales se muestran a continuación (solamente la parte de conexiones, ya que es de lo que se hará uso durante el desarrollo de la tarea).
Y, además, se debe de conocer la estructura de la placa a utilizar, que en este caso fue la Arduino UNO. (recordando que también se puede desarrollar el programa en ESP32).
Y con lo anterior desarrollado, se procede a mostrar el código del programa.

## Código del programa:

```
#include <ESP32Servo.h>

#define servoPin1 10 // Pin para el servo 1

#define servoPin2 9 // Pin para el servo 2

Servo servo1; // Objeto controlador del servo 1

Servo servo2; // Objeto controlador del servo 2

double lar1 = 10; // Largo del servo 1

double lar2 = 6; // Largo del servo 2

double Pi = 3.14159265359;

void menu() {

Serial.println();

Serial.println("Menú");

Serial.println("1. Cinemática Directa");

Serial.println("2. Cinemática Inversa");

Serial.println("Ingresa un número");

Serial.println();

}

void setup() {

// Inicializar el puerto serial
```

```
Serial.begin(115200);
// Inicializar los servos
servo1.attach(servoPin1);
servo2.attach(servoPin2);
// Establecer los ángulos iniciales
servo1.write(90); // Ángulo para servo1 90 grados
servo2.write(0); // Ángulo para servo2 0 grados
menu();
}
bool validaAng(int angulo, int i) {
bool band = false;
if (i == 0) { // Verificar ángulo del servo 1 (-90 a 90 grados)
if (angulo >= -90 && angulo <= 90) {
band = true;
}
} else { // Verificar ángulo del servo 2 (0 a 180 grados)
if (angulo >= 0 && angulo <= 180) {
band = true;
}
}
return band;
}
int angS1(int ang1) {
```

```
int gN = 90;
if (ang1 > 0 && ang1 <= 90) { // Ángulo de 0 a 90 grados
gN = 90 + ang1;
} else if (ang1 < 0 && ang1 >= -90) { // Ángulo de 0 a -90 grados
gN = 90 + ang1;
} else if (ang1 == 0) { // 90 grados se considera 0
gN = 90;
} else { // Regresar 90 si el ángulo se sale de rango
Serial.println("Valor fuera de rango");
gN = 90;
}
return gN;
}
void cDirecta() {
int angulos[2];
int angulo = 200;
int ang1;
int ang2;
bool band = false;
for (int i = 0; i < 2; i++) {
Serial.print("Ingresa el valor del ángulo para el servo ");
Serial.println(i + 1);
delay(5000);
```

```
do {
if (Serial.available() > 0) {
angulo = Serial.readString().toInt();
band = validaAng(angulo, i);
if (band) {
angulos[i] = angulo;
break;
} else {
Serial.print("Ingresa el ángulo ");
Serial.print(i + 1);
Serial.println(" nuevamente");
}
}
} while (band);
Serial.println("Ángulo recibido");
}
ang1 = angulos[0];
ang2 = angulos[1];
Serial.print("Angulo1 = ");
Serial.println(ang1);
Serial.print("Angulo2 = ");
Serial.println(ang2);
int angM = angS1(ang1);
```

```
Serial.print("Moviendo servo1 a ");
Serial.println(ang1);
servo1.write(angM);
delay(2000);
Serial.print("Moviendo servo2 a ");
Serial.println(ang2);
servo2.write(ang2);
float radA1 = (ang1 * Pi) / 180;
float radA2 = (ang2 * Pi) / 180;
Serial.println(radA1);
Serial.println(radA2);
Serial.print("Largo1 = ");
Serial.println(lar1);
Serial.print("Largo2 = ");
Serial.println(lar2);
double y = ((lar1 * cos(radA1)) + (lar2 * cos(radA1 + radA2)))*-1;
double x = ((lar1 * sin(radA1)) + (lar2 * sin(radA1 + radA2)))*-1;
delay(1000);
Serial.print("y = ");
Serial.println(y);
Serial.print("x = ");
Serial.println(x);
menu();
```

```
}
void cInversa() {
String valor = "";
float valores[2];
float x;
float y;
int ch = 88;
for (int i = 0; i < 2; i++) {
Serial.print("Ingresa el valor ");
Serial.println((char)(ch + i));
delay(5000);
do {
if (Serial.available() > 0) {
valor = Serial.readString();
if (valor != "") {
valores[i] = valor.toFloat();
break;
} else {
Serial.print("Ingresa el valor ");
Serial.print(i + 1);
Serial.println(" nuevamente");
}
}
```

```
} while (valor != "");
Serial.println("Valor recibido");
}
x = valores[0];
y = valores[1];
Serial.println("Valores Recibidos");
Serial.print("X = ");
Serial.println(x);
Serial.print("Y = ");
Serial.println(y);
float radA2 = acos((sq(x) + sq(y) - sq(lar1) - sq(lar2)) / (2 * lar1 * lar2));
float radA1 = atan(y / x) - atan((lar2 * sin(radA2)) / (lar1 + lar2 * cos(radA2)));
delay(1000);
float ang1 = (radA1 * 180) / Pi;
float ang2 = (radA2 * 180) / Pi;
float aTh2p = ang1 + 180 + 37;
float aTh2n = ang2 * -1;
Serial.print("Ángulo 1 = ");
if (x < 0) {
Serial.println(ang1 + 180);
} else {
Serial.println(ang1);
}
```

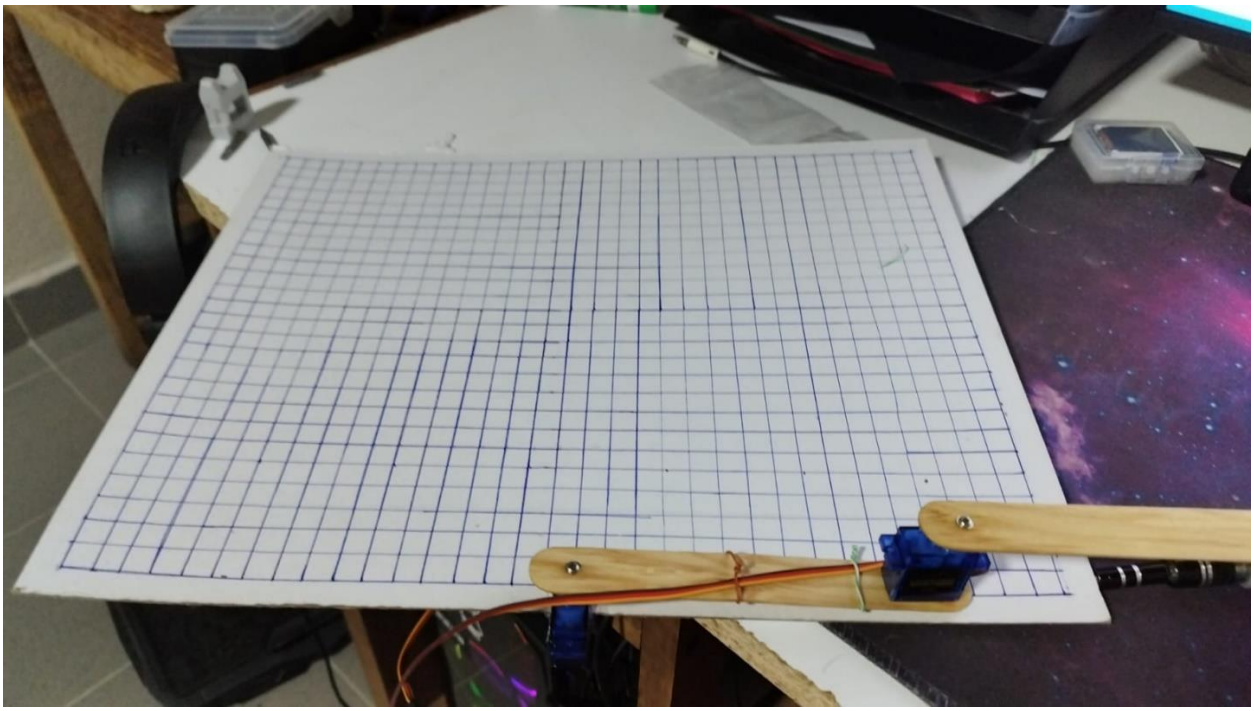
```
Serial.print("Ángulo 2 = ");
Serial.println(ang2);
Serial.print("Th p = ");
Serial.println(aTh2p);
Serial.print("Th n = ");
Serial.println(aTh2n);
if (x < 0) {
servo1.write(ang1 + 180);
} else {
servo1.write(ang1);
}
servo2.write(ang2);
delay(2000);
menu();
}
void loop() {
if (Serial.available()) {
char c = Serial.read();
switch (c) {
case '1':
servo1.write(90);
servo2.write(90);
cDirecta();
```

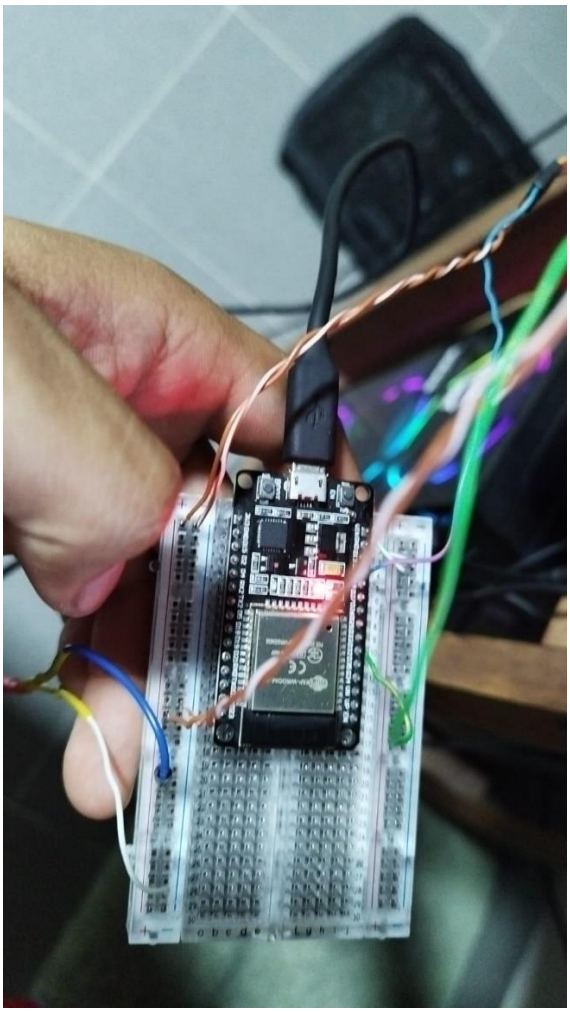
```
break;

case '2':

servo1.write(90);

servo2.write(0);

cInversa();

break;

default:

Serial.println("Opción no válida");

menu();

}

}

}
```

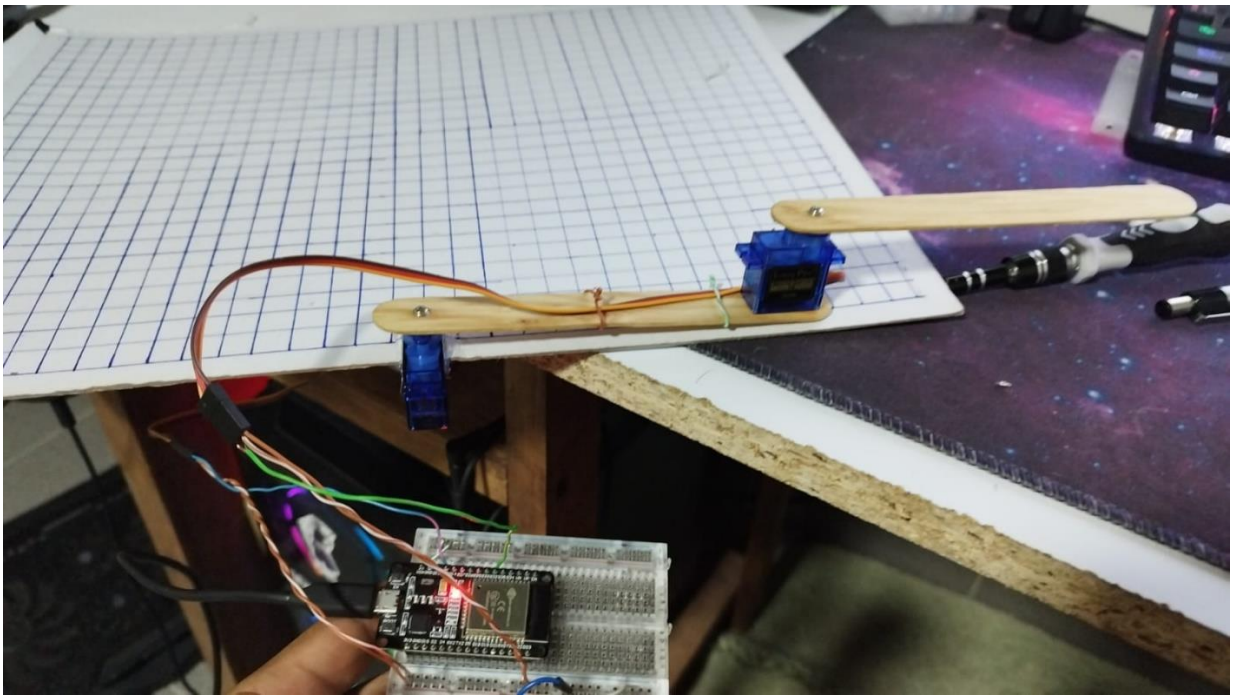
## Funcionamiento del programa:

La función y objetivo principal del programa es mover 2 servomotores en un plano cartesiano que simulan el movimiento de un robot de 2 grados de libertad para indicar la posición del elemento final.

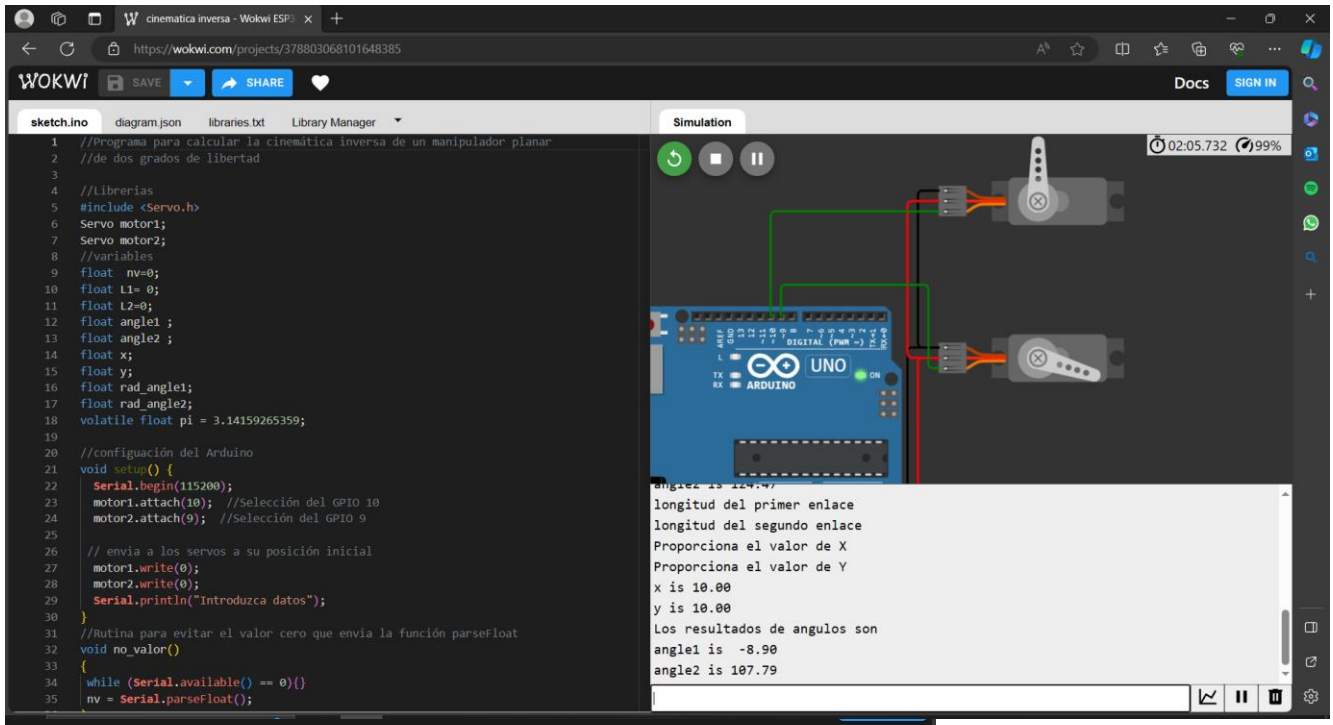
Ahí mismo, se pide al usuario que ingrese la longitud de cada elemento de cada servo (si se llega a ingresar que un elemento mide 0 cm se le pedirá al usuario que vuelva a ingresarlo), y seguido de poner las 2 longitudes, se pedirá que se ingresen el valor de la posición X e Y, a diferencia del programa de cinemática directa, este mostrará los ángulos/grados que se movió cada longitud para poder llegar a la posición deseada. Así mismo, se editó el programa para que, en base a las coordenadas establecidas, se muestren dos posibles soluciones para llegar a dicha posición, y el usuario podrá elegir la que a él le convenga, es decir, seleccionar la que más se apega a una mejor solución.

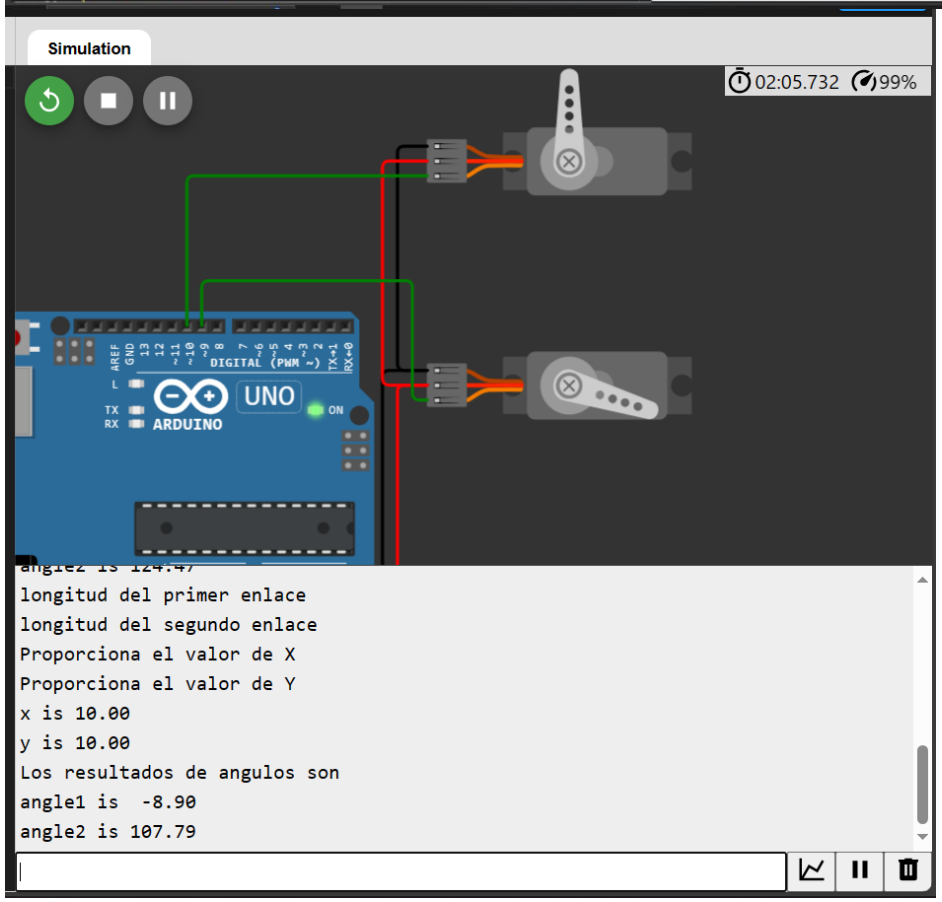




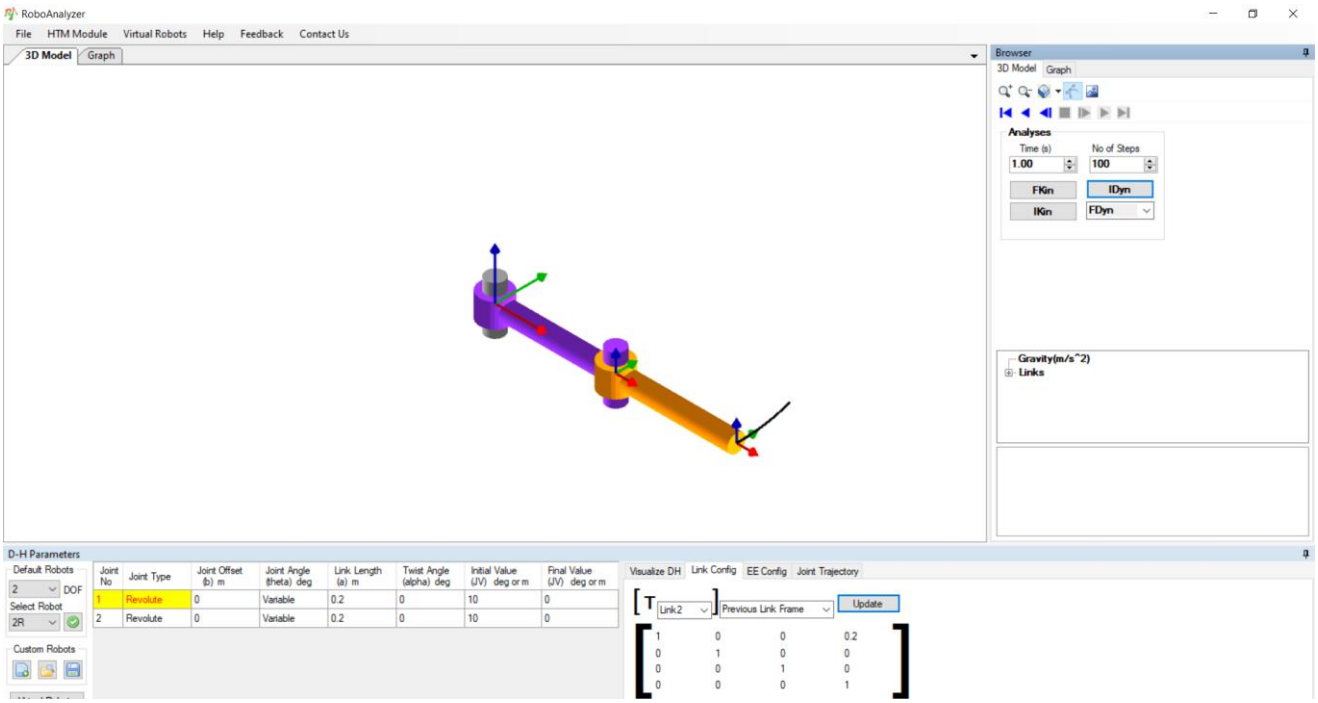


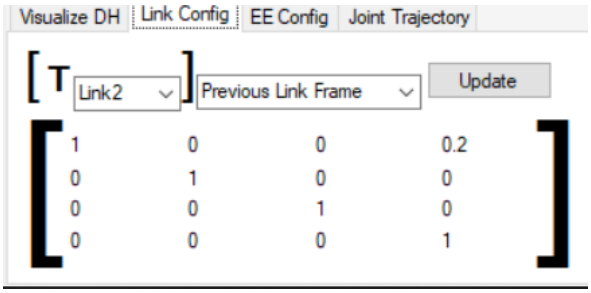
# Simulación en Wokwi

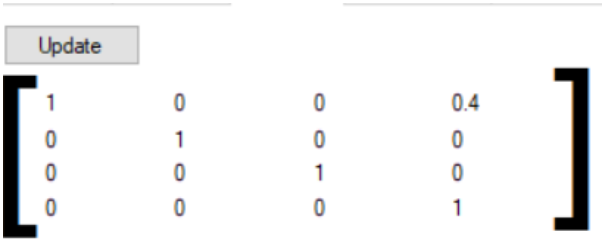


# Simulación de RoboAnalyzer.



## Comentarios:

**Jesus Alejandro Felix Gonzalez**

Durante la elaboración de la práctica de cinemática inversa se logró el entendimiento del uso de coordenadas parea saber los ángulos de la posición final del robot, así como la utilización de esta cinemática dentro de la industria, así como funciona de manera distinta la cinemática directa, el funcionamiento es parecido solo cambia el uso de grados a posiciones en el plano cartesiano.

**Luis Angel García Feregrino**

La cinemática inversa es un área de investigación activa en robótica y se han propuesto numerosos algoritmos y técnicas para mejorar la eficiencia y precisión del cálculo. Una vez calculados los ángulos de las articulaciones mediante la cinemática inversa, se pueden utilizar técnicas de control, como el controlador PID (Proporcional-Integral-Derivativo), para guiar al robot y lograr movimientos suaves y precisos en su espacio de trabajo. Es importante considerar las restricciones físicas y de diseño del robot al calcular la cinemática inversa. Algunas configuraciones de ángulos de articulación pueden no ser alcanzables debido a limitaciones de espacio o colisiones entre los enlaces. La cinemática inversa de un robot con dos grados de libertad implica calcular los ángulos de las articulaciones necesarios para alcanzar una posición y orientación específicas del elemento final. es un proceso matemático complejo que involucra relaciones geométricas y trigonométricas, y puede tener múltiples soluciones. La cinemática inversa es esencial para el control y programación de los robots, permitiendo el movimiento preciso y eficiente en diferentes tareas y entornos.

**Ulises Becerril Valdés**

Dentro de la práctica de cinemática inversa pudimos aprender cómo es que los robots reales funcionan y como es que se implementan según sus grados de libertad, en la industria de los robots, lo que más se usa es la cinemática inversa ya que es de mucha más utilidad dar un punto en el espacio para que sea calculado que ingresar los ángulos en los que se desea mover cada grado de libertad como es el caso de la cinemática directa, pero sin embargo nos encontramos con algunas dificultades dentro del ejercicio de la práctica, esto debido a problemas causaos por el hardware, ya sea por el mal funcionamiento de Arduino o por el desgaste de los servomotores, pese a esto, pudimos concluir la práctica de manera satisfactoria.

**Daisy Montserrat Sánchez Hernández**

En la elaboración de la práctica de cinemática inversa, fue un poco complicado en cuanto a la medición de ángulos ya que teníamos que verificar con distintos intentos si las predicciones de movimiento eran correctas en la realización de nuestra practica con los servomotores para posicionarnos en las coordenadas correctas ya que nos teníamos que ubicar en el plano cartesiano en coordenadas positivas y también negativas, también tuvimos problemas que solucionar con el uso del material que nos proporcionan en laboratorio ya que en nuestro primer intento no funcionó correctamente el Arduino y temíamos que fueran nuestros servomotores

los que ya no sirvieran, El objetivo de la cinemática inversa es encontrar los valores que deben tomar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial.