UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

Modelado y control cinemático

Carlos Mira López
Nicolás Miró Mira
Vittorio Alessandro Esposito Ceballos

Modelado y Control de Robots

4.° curso - Grado en Ingeniería ...

Diciembre de 2025

Índice general

1 Introducción

En este proyecto se diseña y construye un robot manipulador de al menos tres grados de libertad, controlado con servomotores y una placa Arduino UNO. El objetivo es que el robot pueda mover sus articulaciones de manera precisa, calcular la posición del efector final mediante cinemática directa, y determinar los ángulos necesarios para llegar a posiciones deseadas mediante cinemática inversa. Además, se desarrolla una interfaz de usuario que permite controlar el robot, ver su posición en coordenadas articulares y cartesianas, y enviar comandos de movimiento de forma sencilla.

2 Descripción del diseño mecánico y electrónico

Hemos optado por el diseño y ensamblaje de un robot angular con 3 grados de libertad, los cuales aparecen como una rotación en la base en el eje x, un movimiento horizontal en el eje z con respecto a la base gracias a un rodamiento que hemos implementado y un movimiento en el eje x del codo del robot, también gracias a un rodamiento implementado.

Utilizamos 3 servos, uno bajo la base para realizar el giro, otro alineado con el rodamiento del brazo para llevar a cabo el movimiento y otro alineado con el rodamiento del codo, también para su movimiento. En cuanto al cableado cada servomotor se conecta al Arduino mediante tres cables: alimentación (+5V), tierra (GND) y señal. Los pines de señal de cada servo se han conectado a pines digitales del Arduino, mientras que todas las tierras se unifican en común para garantizar un funcionamiento estable.

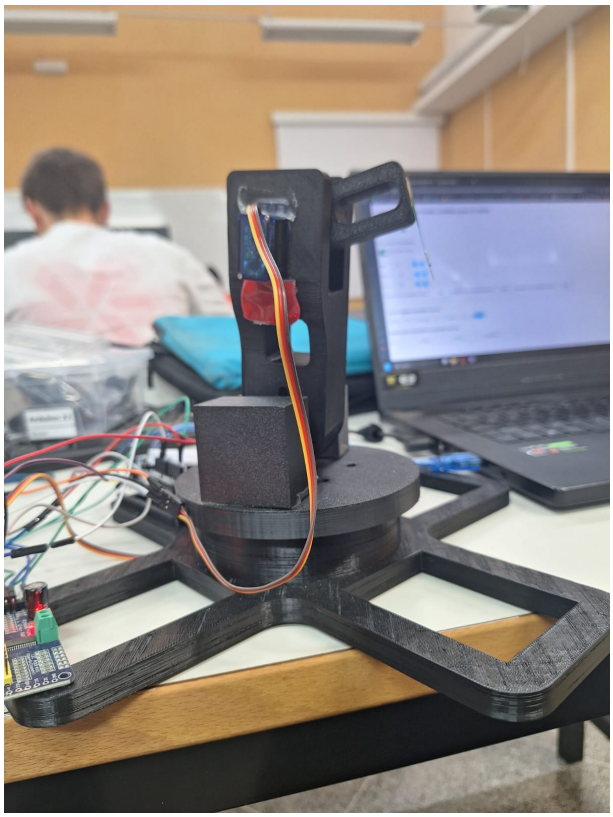

Figura 1: *rebot en plenitud*

Hay varias maneras de definir la posición inicial, nosotros hemos optado por definir una posición conocida como es la [0,0,0], donde la posición del brazo y del codo son horizontales como se muestra en la primera figura, pudiendo calcular así las distancias y facilitar la calibración y el movimiento del robot.

3 Modelo matemático

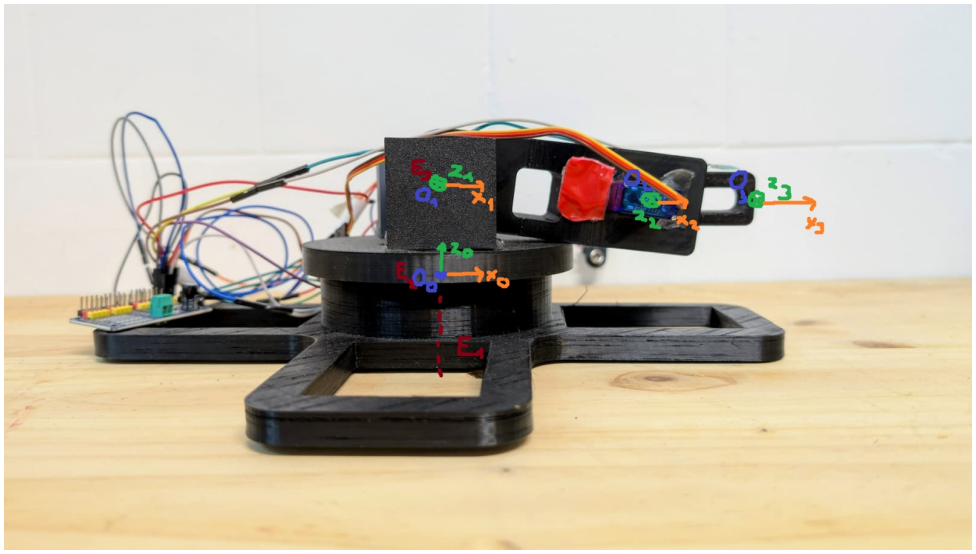

Figura 2: *Ejes del robot*

Para describir la cinemática se emplea la convención de Denavit–Hartenberg (DH). Los parámetros geométricos del robot (según el firmware) son:

d_1 (offset/altura base), d_2 (longitud del primer eslabón), d_3 (longitud del segundo eslabón).

La tabla DH usada para la cinemática directa es:

i	d_i	θ_i	α_i	a_i
1	d_1	θ_1	$\frac{\pi}{2}$	0
2	0	θ_2	0	d_2
3	0	θ_3	0	d_3

En la implementación, los ángulos se obtienen a partir de los servos (en grados) convirtiendo a radianes:

$$\theta_1 = \texttt{baseAngle}\,\frac{\pi}{180}, \qquad \theta_2 = \texttt{brazoAngle}\,\frac{\pi}{180}, \qquad \theta_3 = -\,\texttt{codoAngle}\,\frac{\pi}{180}, \tag{1}$$

donde el signo negativo en θ_3 refleja la inversión mecánica del servo del codo respecto al sentido positivo del modelo.

3.1 Cinemática directa

La cinemática directa permite calcular la posición y orientación del efector final a partir de las articulaciones. En el modelo implementado se consideran 3 grados de libertad (3R):

- **Rotación de base** (θ_1): orienta el brazo hacia una dirección en el plano XY.
- **Rotación de hombro/brazo** (θ_2): eleva o baja el primer eslabón.
- **Rotación de codo** (θ_3): flexiona/extiende el segundo eslabón.

Cada transformación elemental DH ${}^{i-1}T_i$ se construye con la matriz homogénea estándar:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2} $$

Con la tabla anterior, la transformación total es:

$$ {}^{0}T_3 = {}^{0}T_1\,{}^{1}T_2\,{}^{2}T_3. \tag{3} $$

Al multiplicar, la posición del efector final (x, y, z) (columna de traslación de ${}^{0}T_3$) queda:

$$ \boxed{\begin{aligned} x &= \big(d_2\cos\theta_2 + d_3\cos(\theta_2+\theta_3)\big)\cos\theta_1, \\ y &= \big(d_2\cos\theta_2 + d_3\cos(\theta_2+\theta_3)\big)\sin\theta_1, \\ z &= d_1 + d_2\sin\theta_2 + d_3\sin(\theta_2+\theta_3). \end{aligned}} \tag{4} $$

La orientación del efector está dada por la submatriz 3×3 de rotación incluida en ${}^{0}T_3$ (parte superior izquierda de (3)).

3.2 Cinemática inversa

La cinemática inversa determina los ángulos articulares necesarios para alcanzar una posición deseada $p_d = (X, Y, Z)$.

Primero se separa el problema en:

- una rotación de base en el plano XY,
- un problema planar 2R en el plano (r, z), con $r = \sqrt{X^2+Y^2}$.

1) Rotación de base: Con la siguiente ecuación

$$ \boxed{\theta_1 = \operatorname{atan2}(Y, X)}. \tag{5} $$

2) Reducción al plano del brazo.

$$ \boxed{r = \sqrt{X^2+Y^2}}, \qquad \boxed{z = Z - d_1}. \tag{6} $$

3) Ley del coseno para el codo. Definiendo

$$ \boxed{D = \frac{r^2 + z^2 - d_2^2 - d_3^2}{2d_2d_3}}, \tag{7} $$

se tiene $D = \cos\theta_3$. Para que exista solución geométrica debe cumplirse:

$$ \boxed{|D| \le 1}. \tag{8} $$

4) Dos soluciones posibles (codo arriba / codo abajo).

$$\boxed{\theta_3 = \operatorname{atan2}\Big(\pm \sqrt{1 - D^2},\ D \Big)}. \tag{9}$$

El signo $\pm$ determina la rama de solución (por ejemplo, codo arriba o codo abajo).

5) Cálculo del hombro.

$$\boxed{\theta_2 = \operatorname{atan2}(z, r) - \operatorname{atan2}\Big(d_3 \sin\theta_3,\ d_2 + d_3 \cos\theta_3 \Big)}. \tag{10}$$

6) Conversión a grados y adaptación a servos.
En el control real (servos), se pasa a grado

$$\theta[\text{deg}] = \theta[\text{rad}]\, \frac{180}{\pi},$$

y se aplican límites físicos típicos:

$$\theta_{\text{servo}} \in [0^\circ, 180^\circ].$$

Además, para ser coherente con la cinemática directa implementada en el firmware, el codo se invierte:

$$\boxed{\theta_{3,\text{servo}} = -\,\theta_3 \ \ (\text{en grados})}. \tag{11}$$

Finalmente, se deben considerar restricciones mecánicas (límites de giro y colisiones) y la existencia de múltiples soluciones por 9, eligiendo la rama que sea físicamente realizable para el robot.

4 Código e interfaz

4.1 Código implementado en Arduino

Listing 1: *Configuración inicial y variables globales.*

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <math.h>

%% Controlador PCA9685
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

%% Canales para cada servo en el PCA9685
#define SERVO_BASE 0
#define SERVO_BRAZO 1
#define SERVO_CODO 2

%% Matriz DH cinematica directa [alpha, a, d, theta]
float Matriz_DH_CD[3][4];
const int N = 3;
const int N1 = 4;

%% Matriz resultado
float res_CN[N1][N1];

%% Angulos actuales
int baseAngle = 0;
int brazoAngle = 0;
int codoAngle = 0;
```

```
%% Distancia entre ejes
float d1 = 0.075;
float d2 = 0.082;
float d3 = 0.050;

%% Resultados de la cinematica inversa
float theta1, theta2, theta3;
const int step = 5;
int modo = 0; %% 0 = directa, 1 = inversa

%% Mapear angulo a pulso
int angleToPulse(int ang) {
    int pulsoMin = 150; %% 0 grados
    int pulsoMax = 600; %% 180 grados
    return map(ang, 0, 180, pulsoMin, pulsoMax);
}

%% Wrapper para mover servo
void moverServo(int canal, int angulo) {
    int pulso = angleToPulse(angulo);
    pwm.setPWM(canal, 0, pulso);
}
```

Listing 2: *Función de Cinemática Inversa (ik_RRR).*

```
%% Función de cinemática inversa RRR
bool ik_RRR(float X, float Y, float Z) {
    const float a2 = d2;
    const float a3 = d3;
    const float h  = d1;

    float th1 = atan2f(Y, X);

    float r = sqrtf(X*X + Y*Y);
    float z = Z - h;

    float D = (r*r + z*z - a2*a2 - a3*a3) / (2.0f * a2 * a3);

    %% Verificación de dominio matemático
    if (D > 1.0f) D = 1.0f;
    if (D < -1.0f) D = -1.0f;
    if (isnan(D)) return false;

    float s3 = sqrtf(fmaxf(0.0f, 1.0f - D*D));
    float th3_geom = atan2f(-s3, D);   %% codo abajo

    float num = a3 * sinf(th3_geom);
    float den = a2 + a3 * cosf(th3_geom);
    float th2_geom = atan2f(z, r) - atan2f(num, den);

    const float RAD2DEG = 180.0f / PI;
    float g1 = th1       * RAD2DEG;
    float g2 = th2_geom  * RAD2DEG;
    float g3 = th3_geom  * RAD2DEG;

    %% Normalización de ángulos
    if (g1 < 0.0f) g1 += 360.0f;
    if (g1 > 180.0f) g1 = 360.0f - g1;

    float servo_base  = g1;
    float servo_brazo = g2;
    float servo_codo  = -g3;
```

```
    %% Restricción de rango físico
    servo_base  = constrain(servo_base,  0.0f, 180.0f);
    servo_brazo = constrain(servo_brazo, 0.0f, 180.0f);
    servo_codo  = constrain(servo_codo,  0.0f, 180.0f);

    theta1 = servo_base;
    theta2 = servo_brazo;
    theta3 = servo_codo;

    return true;
}
```

Listing 3: *Funciones para manejo de matrices y DH.*

```
%% Inicializar matriz DH
void inicializar_matriz(){
    Matriz_DH_CD[0][0] =  M_PI/2;
    Matriz_DH_CD[0][1] =  0.0;
    Matriz_DH_CD[0][2] =  d1;
    Matriz_DH_CD[0][3] =  0.0;

    Matriz_DH_CD[1][0] =  0.0;
    Matriz_DH_CD[1][1] =  d2;
    Matriz_DH_CD[1][2] =  0.0;
    Matriz_DH_CD[1][3] =  0.0;

    Matriz_DH_CD[2][0] =  0.0;
    Matriz_DH_CD[2][1] =  d3;
    Matriz_DH_CD[2][2] =  0.0;
    Matriz_DH_CD[2][3] =  0.0;
}

%% Convertir parámetros DH en matriz homogénea
void dh_to_T(float DH[4][4], float alpha, float a, float d, float theta){
    float ca = cos(alpha);
    float sa = sin(alpha);
    float ct = cos(theta);
    float st = sin(theta);

    DH[0][0] = ct;       DH[0][1] = -st * ca;    DH[0][2] = st * sa;     DH[0][3] = a * ct;
    DH[1][0] = st;       DH[1][1] =  ct * ca;    DH[1][2] = -ct * sa;    DH[1][3] = a * st;
    DH[2][0] = 0.0;      DH[2][1] = sa;          DH[2][2] = ca;          DH[2][3] = d;
    DH[3][0] = 0.0;      DH[3][1] = 0.0;         DH[3][2] = 0.0;         DH[3][3] = 1.0;
}

%% Multiplicar matrices 4x4
void multiplicarMatrices(float A[N1][N1], float B[N1][N1], float C[N1][N1]) {
    for (int i = 0; i < N1; i++) {
        for (int j = 0; j < N1; j++) {
            C[i][j] = 0;
            for (int k = 0; k < N1; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
```

Listing 4: *Cálculo de la Cinemática Directa.*

```
%% Calcular cinemática directa
void calcular_directa(){
```

```
    float alpha[N], a[N], d[N], theta[N];
    float A1[N1][N1], A2[N1][N1], A3[N1][N1];
    float T0[N1][N1], T1[N1][N1];

    for (int i = 0; i < N; i++) {
        alpha[i] = Matriz_DH_CD[i][0];
        a[i]     = Matriz_DH_CD[i][1];
        d[i]     = Matriz_DH_CD[i][2];
        theta[i] = Matriz_DH_CD[i][3];
    }

    theta[0] += baseAngle * M_PI / 180.0;
    theta[1] += brazoAngle * M_PI / 180.0;
    theta[2] += -codoAngle * M_PI / 180.0;

    dh_to_T(A1, alpha[0], a[0], d[0], theta[0]);
    dh_to_T(A2, alpha[1], a[1], d[1], theta[1]);
    dh_to_T(A3, alpha[2], a[2], d[2], theta[2]);

    float identidad[N1][N1] = {
        {1.0, 0.0, 0.0, 0.0},
        {0.0, 1.0, 0.0, 0.0},
        {0.0, 0.0, 1.0, 0.0},
        {0.0, 0.0, 0.0, 1.0}
    };

    multiplicarMatrices(identidad, A1, T0);
    multiplicarMatrices(T0, A2, T1);
    multiplicarMatrices(T1, A3, res_CN);
}
```

Listing 5: *Configuración (Setup) y Bucle Principal (Loop).*

```
%% Configuración inicial
void setup() {
    Serial.begin(9600);
    pwm.begin();
    pwm.setPWMFreq(60);

    moverServo(SERVO_BASE, baseAngle);
    moverServo(SERVO_BRAZO, brazoAngle);
    moverServo(SERVO_CODO, codoAngle);

    inicializar_matriz();
}

%% Bucle principal
void loop() {
    if (Serial.available()) {
        String cmd = Serial.readStringUntil('\n');
        cmd.trim();

        if (cmd == "0") {
            modo = 0;
        }
        else if (cmd == "1") {
            modo = 1;
        }
        %% Modo inverso: SET,X,Y,Z
        else if (cmd.startsWith("SET,") && modo == 1) {
            cmd.remove(0, 4);
            int coma1 = cmd.indexOf(',');
            int coma2 = cmd.lastIndexOf(',');
```

```
            if (coma1 > 0 && coma2 > coma1) {
                float b = cmd.substring(0, coma1).toFloat();
                float r = cmd.substring(coma1 + 1, coma2).toFloat();
                float c = cmd.substring(coma2 + 1).toFloat();

                bool func = ik_RRR(b, r, c);

                if (func == true){
                    moverServo(SERVO_BASE, theta1);
                    moverServo(SERVO_BRAZO, theta2);
                    moverServo(SERVO_CODO, theta3);
                }
                baseAngle = theta1;
                brazoAngle = theta2;
                codoAngle = theta3;
            }
        }
        %% Controles manuales
        else if (cmd == "Base+" && modo == 0) {
            baseAngle = constrain(baseAngle + step, 0, 180);
            moverServo(SERVO_BASE, baseAngle);
        }
        else if (cmd == "Base-" && modo == 0) {
            baseAngle = constrain(baseAngle - step, 0, 180);
            moverServo(SERVO_BASE, baseAngle);
        }
        else if (cmd == "Brazo+" && modo == 0) {
            brazoAngle = constrain(brazoAngle + step, 0, 180);
            moverServo(SERVO_BRAZO, brazoAngle);
        }
        else if (cmd == "Brazo-" && modo == 0) {
            brazoAngle = constrain(brazoAngle - step, 0, 180);
            moverServo(SERVO_BRAZO, brazoAngle);
        }
        else if (cmd == "Codo+" && modo == 0) {
            codoAngle = constrain(codoAngle + step, 0, 180);
            moverServo(SERVO_CODO, codoAngle);
        }
        else if (cmd == "Codo-" && modo == 0) {
            codoAngle = constrain(codoAngle - step, 0, 180);
            moverServo(SERVO_CODO, codoAngle);
        }
    }

    calcular_directa();

    Serial.print(baseAngle);
    Serial.print(",");
    Serial.print(brazoAngle);
    Serial.print(",");
    Serial.print(codoAngle);
    Serial.print(",");
    Serial.print(res_CN[0][3]);
    Serial.print(",");
    Serial.print(res_CN[1][3]);
    Serial.print(",");
    Serial.print(res_CN[2][3]);
    Serial.print("\n");

    delay(500);
}
```

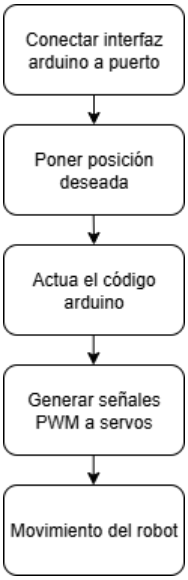

Figura 3: *Diagrama de funcionamiento*

4.2 Interfaz de usuario

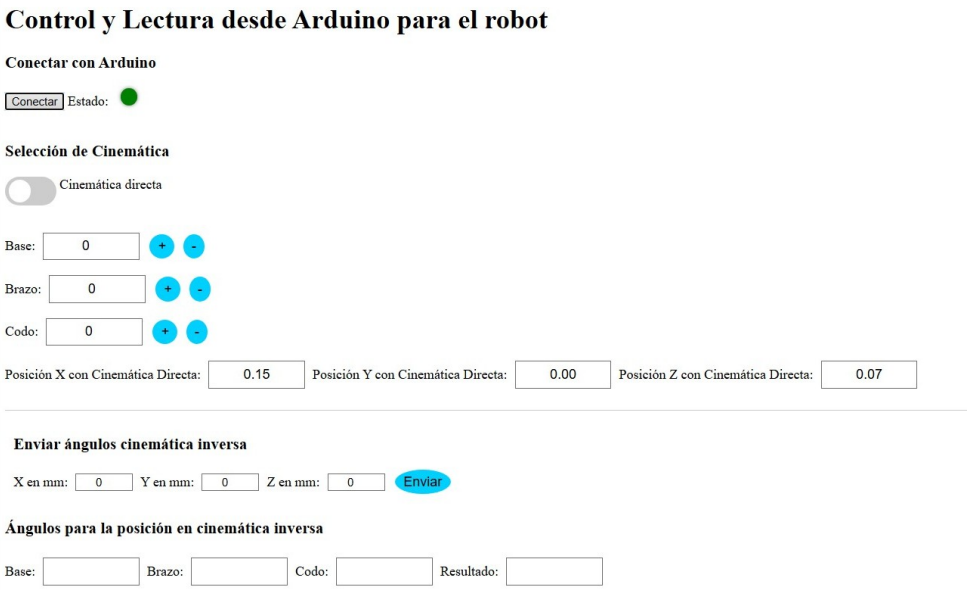

Figura 4: *Interfaz de usuario*

Esta es la interfaz de usuario que hemos diseñado, tenemos en primera parte un botón para conectar con el robot, junto con un icono de color que nos indica visualmente si la conección ha sido exitosa o no. debajo de eso elegimos el tipo te cinemática, si la opción está pulsada es cinemática directa, en caso contrario es cinemática inversa; debajo de eso tenemos cajas de texto ajustables para base, brazo y codo del robot, todas junto a iconos de + y -, donde podemos mover paso a paso individualmente cada parte en incrementos o decrementos de 5 grados, acompañado con eso debajo cajas de texto que indican las posición del robot en los ejes X Y y Z.

En cuanto a la parte de la cinemática inversa tenemos cajas de texto donde modificamos el valor de las posiciones de X,Y y Z medidas en milímetros, a lo que después presionamos el botón

enviar, lo que mueve el robot a las posiciones elegidas, mostrando en las cajas de abajo el ángulo en la base el brazo y el codo, además del resultado del cálculo de la cinemática inversa

5 Pruebas y demostración

En esta sección presentaremos pruebas del funcionamiento del robot

Link a video de cinemática directa

Link a video de cinemática inversa

y este es el resultado final de funcionamiento conjunto:

Video final