

Índice general

1. Documento I: Memoria Descriptiva	1
1.1. Información previa	2
1.1.1. Introducción	2
1.1.2. Peticionario	3
1.1.3. Antecedentes	3
1.1.4. Ubicación	4
1.1.5. Necesidades a satisfacer y justificación	4
1.1.6. Objeto del trabajo	5
1.1.7. Definiciones y abreviaturas	6
1.1.8. Marco teórico	7
1.1.9. Tareas realizadas para cumplir el objeto	13
1.1.10. Normas y referencias	14
1.2. Descripción del trabajo realizado	15
1.2.1. Visión general de la solución adoptada	15
1.2.2. Diagrama de flujo del proyecto	16
1.2.3. Requisitos de diseño	19
1.2.4. Hardware y software utilizados	20
1.2.5. Dificultades encontradas y soluciones adoptadas	28
1.2.6. Disposición física de los elementos	31
1.2.7. Descripción detallada de la solución	32
1.2.8. Resultados	33
1.2.9. Conclusiones finales	33
1.2.10. Posibles mejoras y trabajo futuro	33
2. Documento II: Memoria de Cálculo	34
Documento II: Memoria de Cálculo	34
2.1. Hipótesis, condiciones de contorno y criterios de diseño	34
2.2. Modelos matemáticos y ecuaciones empleadas	34
2.3. Procedimientos de cálculo y dimensionamiento	34
2.4. Resultados de cálculo (tablas y figuras)	34

2.5. Verificación, validación y análisis de incertidumbre	34
2.6. Conclusiones de la memoria de cálculo	34
3. Documento III: Pliego de Condiciones Técnicas	35
Documento III: Pliego de Condiciones Técnicas	35
3.1. Condiciones generales	35
3.2. Clasificación de la instalación	35
3.3. Criterios de diseño y ejecución	35
3.4. Materiales y equipos	35
3.5. Condiciones de montaje	35
3.6. Programa de mantenimiento	35
3.7. Pruebas y puesta en servicio	35
3.8. Documentación necesaria	35
4. Documento IV: Presupuesto	36
Documento IV: Presupuesto	36
4.1. Cuadro de precios unitarios	36
4.2. Medición y valoración	36
4.3. Resumen por capítulos	36
4.4. Presupuesto de ejecución material	36
4.5. Costes indirectos y gastos generales	36
4.6. Presupuesto total	36
5. Documento V: Esquemas y Conexionados	37
Documento V: Esquemas y Conexionados	37
5.1. Diagrama de bloques del sistema	37
5.2. Esquemas de comunicaciones / red	37
6. Documento VI: Anexos	38
Documento VI: Anexos	38
6.1. Listados de código y scripts	38
6.2. Manuales / datasheets relevantes	38

Índice de figuras

1.1.	Ubicación del laboratorio donde se desarrolló el proyecto.	4
1.2.	Rotación entre sistemas de referencia. De Barrientos et al. [1].	9
1.3.	Transformación homogénea entre sistemas de referencia. De Barrientos et al. [1].	11
1.4.	Modelo 3D del UR5 y su modelo cinemático correspondiente. De Research Gate et al. [12].	12
1.5.	Diagrama de flujo del proyecto.	18
1.6.	Brazo robótico UR5 de Universal Robots de [5].	20
1.7.	Tableta gráfica Wacom Intuos Pro Large de [13].	21
1.8.	Tableta gráfica Wacom IOTP de [14].	22
1.9.	Sensor de movimiento 3D mediante potenciómetros y robot.	23
1.10.	Entorno de desarrollo PyCharm Community Edition 2021.2.3.	24
1.11.	Entorno de desarrollo MATLAB R2023b.	25
1.12.	Software de captura de datos desde la tableta Wacom WIPL.	26
1.13.	Software de captura de datos desde la tableta Wacom IOTP.	27
1.14.	Entorno de desarrollo Arduino IDE.	27
1.15.	Ejemplo de pico de velocidad generado por ausencia de puntos intermedios al separar demasiado el lápiz de la tableta.	28
1.16.	Lápiz digital de la tableta Wacom WIPL, con los botones laterales que pueden activarse involuntariamente durante la escritura.	29
1.17.	Valores anómalos de la coordenada Z asociados a la activación involuntaria de los botones del lápiz, que generan distorsiones en la señal de velocidad.	29
1.18.	Ejemplo de desfase entre la trayectoria R y la trayectoria RT.	30
1.19.	Esquema de la disposición física del sistema.	31
1.20.	Disposición real del sistema en el laboratorio.	32
1.21.	Flujo de funcionamiento del sistema captor 3D (vertical).	32
1.22.	Estructura jerárquica de los directorios de la base de datos	32

Índice de cuadros

1.1. Estructura de los ficheros de datos por palabra.	33
---	----

Capítulo 1

Documento I: Memoria Descriptiva

1.1. Información previa

1.1.1. Introducción

En la actualidad, los sistemas de digitalización han adquirido una gran relevancia en numerosos ámbitos científicos y tecnológicos. Dispositivos como las tabletas gráficas permiten registrar de manera precisa trazos y movimientos, lo que abre la puerta a múltiples aplicaciones que van desde el diseño asistido hasta el análisis del comportamiento humano. Sin embargo, todo instrumento de medida presenta un margen de error, y conocer dicho error resulta esencial para valorar la fiabilidad de los datos recogidos y garantizar la validez de los estudios que se apoyan en ellos.

El análisis de la escritura constituye un campo especialmente interesante dentro de este marco, ya que la forma en la que una persona escribe puede ofrecer información sobre sus capacidades motoras, cognitivas o incluso sobre posibles alteraciones neurológicas. Por ello, disponer de un sistema que permita comparar la escritura humana con la reproducida por un robot o capturada por distintos dispositivos digitalizadores ofrece una oportunidad única para estudiar cómo se generan los trazos y qué diferencias se introducen por efecto del medio utilizado.

Además de su interés puramente técnico, la posibilidad de caracterizar y cuantificar estos errores abre nuevas líneas de aplicación en el desarrollo de herramientas basadas en inteligencia artificial. Sistemas entrenados con este tipo de datos pueden contribuir a la detección temprana de enfermedades, al análisis del envejecimiento motor o a la mejora de interfaces hombre-máquina en el ámbito médico y educativo. Así, el estudio de los errores de digitalización no debe entenderse únicamente como un ejercicio académico, sino como un paso necesario hacia el aprovechamiento de la escritura como fuente de información biomédica.

El presente trabajo se centra en la comparación de distintos métodos de registro y reproducción de la escritura, empleando tanto la acción directa de un usuario sobre una tableta como la ejecución de trazos mediante un robot colaborativo. El proceso seguido consiste, a grandes rasgos, en capturar un conjunto de palabras, reproducirlas en diferentes sistemas y analizar las diferencias existentes entre los resultados. A partir de dichas comparaciones es posible estimar el error asociado a cada dispositivo y obtener conclusiones sobre la fiabilidad de los datos que generan.

En definitiva, este estudio pretende aportar una visión global sobre la importancia de conocer las limitaciones de los sistemas de digitalización, destacando su potencial como herramienta para el análisis de la escritura y su aplicación futura en el entrenamiento de algoritmos de inteligencia artificial destinados al ámbito médico y al desarrollo de tecnologías de apoyo.

1.1.2. Peticionario

El presente proyecto ha sido solicitado por el [Instituto para el Desarrollo Tecnológico y la Innovación en Comunicaciones \(IDeTIC\)](#), en el marco de las actividades académicas vinculadas al Grado en Ingeniería Electrónica Industrial y Automática de la Escuela de Ingenieros Industriales y Civiles de la Universidad de Las Palmas de Gran Canaria.

1.1.3. Antecedentes

La tecnología de captura de movimiento permite registrar movimientos físicos y digitalizarlos con el fin de analizarlos o replicarlos en entornos digitales. A pesar de los avances alcanzados en los últimos años, estos sistemas todavía enfrentan un reto importante: su precisión y fiabilidad siguen siendo insuficientes para reproducir con exactitud los movimientos humanos mediante un brazo robótico.

El uso de técnicas de captura de movimiento no es reciente. Sus primeras aplicaciones se dieron en el ámbito de la animación digital y la industria del entretenimiento, donde se empleaban para trasladar los gestos de actores a personajes virtuales. Con el paso del tiempo, la tecnología se ha extendido a sectores tan diversos como la ingeniería, la medicina, el deporte y la robótica, en los que la necesidad de registrar trayectorias con precisión ha cobrado mayor relevancia.

Los instrumentos de digitalización de movimiento, como las tabletas gráficas y los sensores de movimiento tridimensional (3D), se han consolidado como herramientas fundamentales en ámbitos como la robótica, la animación digital y la medicina. Estos dispositivos proporcionan a los profesionales la capacidad de capturar, manipular y analizar movimientos de manera más eficiente y precisa que mediante técnicas tradicionales.

En particular, las tabletas gráficas permiten registrar movimientos en dos dimensiones. Son dispositivos relativamente accesibles y fáciles de utilizar, aunque su nivel de precisión depende en gran medida de la calidad y la gama del producto. Por otra parte, los sensores de movimiento 3D ofrecen la posibilidad de capturar desplazamientos en un espacio tridimensional, aportando una representación más realista y completa, si bien presentan un coste de adquisición elevado y una mayor complejidad técnica en su uso e integración.

Más allá de las diferencias entre dispositivos, existen limitaciones comunes en este tipo de tecnologías: la necesidad de calibración frecuente, la influencia del ruido en la señal, la latencia en la transmisión de datos y la dependencia del hardware utilizado. Estas restricciones pueden comprometer la fiabilidad del sistema, especialmente en aplicaciones donde se requiere alta precisión, como en la interacción humano–robot o la programación automática de trayectorias.

En este contexto, el presente proyecto plantea un análisis comparativo de los errores generados por diferentes instrumentos digitalizadores. El propósito es comprender cómo dichos errores afectan a la precisión y fiabilidad de los sistemas de captura, ya que es-

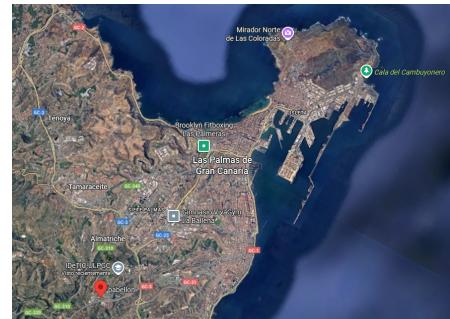
tos aspectos son determinantes para el correcto funcionamiento de aplicaciones en las que la exactitud del movimiento resulta crítica, como la interacción humano–robot, la rehabilitación asistida por robot o la simulación de procesos en entornos virtuales.

1.1.4. Ubicación

El proyecto se llevó a cabo en el [TODO: Nombre del laboratorio] del IDeTIC. En la Figure 1.1a se muestra la ubicación del pabellón donde se sitúa el laboratorio dentro del campus universitario, mientras que en la Figure 1.1b se indica su localización en el municipio de Las Palmas de Gran Canaria.



(a) Ubicación del laboratorio dentro del campus universitario.



(b) Ubicación del laboratorio en el municipio de Las Palmas.

Figura 1.1: Ubicación del laboratorio donde se desarrolló el proyecto.

1.1.5. Necesidades a satisfacer y justificación

El desarrollo del presente proyecto responde a la necesidad de contar con herramientas que permitan evaluar con rigor la precisión y fiabilidad de los sistemas de captura de movimiento empleados en la interacción con brazos robóticos. A partir del análisis de los antecedentes, se identifican las siguientes necesidades principales:

- **Medir objetivamente los errores de digitalización:** disponer de un procedimiento que cuantifique la diferencia entre el movimiento humano, el ejecutado por el brazo robótico y el registrado por los instrumentos de captura.
- **Comparar distintas tecnologías de captura:** analizar el comportamiento de tabletas gráficas de diferentes gamas frente a sensores tridimensionales, con el fin de establecer criterios de selección en función de la relación coste–prestaciones.
- **Mejorar la fiabilidad de la interacción humano–robots:** detectar las limitaciones de cada sistema de captura que puedan afectar a la ejecución precisa de trayectorias y proponer estrategias de mitigación.

- **Facilitar la integración en entornos académicos e industriales:** generar una base de conocimiento que permita seleccionar tecnologías de digitalización adecuadas para proyectos de investigación, docencia y aplicaciones prácticas en automatización y robótica.

En conjunto, estas necesidades reflejan la importancia de evaluar de forma crítica las herramientas de captura de movimiento y su impacto en la programación y control de robots, garantizando resultados reproducibles y útiles para futuras aplicaciones en interacción humano–robot.

1.1.6. Objeto del trabajo

El objeto del presente proyecto es el estudio, caracterización y análisis de los errores producidos por distintos instrumentos digitalizadores de captura de movimiento cuando un brazo robótico ejecuta trayectorias previamente programadas.

El trabajo persigue comprender de forma detallada cómo se comportan diferentes sistemas de adquisición de movimiento frente a la ejecución real del robot y frente al gesto humano que origina la trayectoria. Para ello, el análisis se centra en tres tipos de error principales:

1. El error existente entre el movimiento realizado por el usuario y el que finalmente ejecuta el brazo robótico.
2. El error entre el movimiento ejecutado por el brazo robótico y el que registran los instrumentos digitalizadores.
3. El error resultante de comparar directamente el movimiento realizado por el usuario con el registrado por los instrumentos digitalizadores.

Como instrumentos de digitalización se emplearán tabletas gráficas de distintas gamas, seleccionadas en función de su precio y calidad, con el fin de estudiar cómo estas variables influyen en la precisión de los datos obtenidos. Además, se incorporarán sensores de captura de movimiento tridimensional (3D), que permitirán contrastar y complementar los resultados derivados de las tabletas, ofreciendo una perspectiva más amplia y comparativa de las capacidades de cada tecnología.

Este enfoque permitirá no solo cuantificar los errores presentes en cada etapa (Usuario–Robot, Robot–Instrumento, Usuario–Instrumento), sino también identificar las limitaciones y fortalezas de cada sistema de digitalización. Con ello se busca establecer una base objetiva que facilite la selección de la tecnología más adecuada en función de los requisitos de precisión, coste y aplicabilidad en distintos entornos de trabajo, especialmente en el ámbito de la interacción humano–robot y la programación de trayectorias asistidas.

1.1.7. Definiciones y abreviaturas

Definiciones

Cinemática Rama de la mecánica que estudia el movimiento de los cuerpos sin considerar las fuerzas que lo producen. En robótica, se refiere al análisis del movimiento de los manipuladores y sus eslabones..

Matlab Entorno de programación y cálculo numérico desarrollado por MathWorks, ampliamente utilizado en ingeniería y ciencias para el análisis de datos, simulación y desarrollo de algoritmos..

Python Lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su sintaxis clara y legibilidad, utilizado en una amplia variedad de aplicaciones, desde desarrollo web hasta análisis de datos y aprendizaje automático..

Red Neuronal Modelo computacional inspirado en la estructura y funcionamiento del cerebro humano, compuesto por nodos (neuronas) interconectados que procesan información y aprenden a partir de datos..

Robot Máquina programable capaz de realizar tareas de forma autónoma o semiautónoma, generalmente mediante la ejecución de movimientos precisos y repetitivos..

Acrónimos

CSV Archivo de valores separados por comas (*Comma-Separated Values*).

HTR Error entre el movimiento humano y el ejecutado por el robot.

HTRT Error entre el movimiento humano y el registrado por la tableta.

IDE Entorno de Desarrollo Integrado (*Integrated Development Environment*).

IDeTIC Instituto para el Desarrollo Tecnológico y la Innovación en Comunicaciones.

IOTP Wacom IOT Paper 3.

MAE Mean Absolute Error.

MSE Mean Squared Error.

MTP Media Transfer Protocol.

R Robot.

RMSE Root Mean Square Error.

RRT Error entre el movimiento ejecutado por el robot y el registrado por la tableta.

RT Robot-Tableta.

UR5 Universal Robots 5.

WIPL Wacom Intuos Pro Large.

1.1.8. Marco teórico

El marco teórico de este proyecto se fundamenta en los principios y conceptos básicos de la robótica, en particular aquellos relacionados con la [Cinemática](#), la dinámica y el control de manipuladores robóticos. Como referencia principal se emplea el libro *Fundamentos de Robótica* de Barrientos et al. [1], en el cual se abordan de manera sistemática los aspectos esenciales de la robótica moderna.

De acuerdo con Barrientos et al. [1], un manipulador robótico puede describirse como un conjunto de eslabones articulados que permiten ejecutar movimientos en el espacio tridimensional mediante la aplicación de leyes de [Cinemática](#) y dinámica. En este contexto, la caracterización de la posición, la orientación y la trayectoria del efecto final resulta fundamental para garantizar la precisión y repetibilidad del sistema.

Asimismo, el estudio de los sistemas de referencia, las transformaciones homogéneas y los algoritmos de control asociados es imprescindible para comprender cómo el [robot](#) interpreta y ejecuta las trayectorias programadas. Estos conceptos se aplicarán directamente en el presente proyecto para analizar la diferencia entre el movimiento humano de referencia, el movimiento ejecutado por el brazo robótico y el movimiento registrado por los distintos instrumentos digitalizadores.

De este modo, el marco teórico no solo proporciona las bases matemáticas y conceptuales de la robótica, sino que también sustenta la motivación del proyecto: evaluar la fiabilidad y precisión de diferentes instrumentos de captura de movimiento en la interacción con un manipulador robótico.

Traslación entre sistemas de referencia

En robótica, el uso de sistemas de referencia es fundamental para describir la posición y el movimiento de un manipulador en el espacio. Cuando se trabaja con más de un sistema de coordenadas, resulta necesario establecer relaciones que permitan expresar un punto definido en un sistema en términos de otro.

La traslación entre sistemas de referencia consiste en desplazar el origen de un sistema con respecto a otro, manteniendo la orientación de sus ejes paralela. Matemáticamente, si se considera un sistema de referencia $\{A\}$ y otro sistema $\{B\}$ cuyo origen se encuentra en

el vector $\mathbf{p} = [p_x, p_y, p_z]^T$ respecto a $\{A\}$, cualquier punto \mathbf{r}_B expresado en coordenadas del sistema $\{B\}$ puede representarse en el sistema $\{A\}$ como la ecuación (1.1):

$$\mathbf{r}_A = \mathbf{r}_B + \mathbf{p} \quad (1.1)$$

donde \mathbf{r}_A es el vector de posición del punto en el sistema $\{A\}$. Esta expresión refleja que la traslación se implementa mediante la suma del vector que define el desplazamiento entre los orígenes.

En términos matriciales, la operación puede escribirse como la ecuación (1.2):

$$\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad (1.2)$$

De esta forma, la traslación permite relacionar de manera directa dos sistemas de referencia que comparten la misma orientación, siendo esta una herramienta básica para la modelización de manipuladores robóticos y el cálculo de trayectorias.

Tal y como señala Barrientos et al. [1], esta formulación es el punto de partida para operaciones más complejas que incluyen además rotaciones y, en general, transformaciones homogéneas, necesarias para describir la [Cinemática de robots](#) manipuladores.

Rotación entre sistemas de referencia

Cuando dos sistemas de coordenadas comparten el mismo origen, pero presentan una orientación diferente de sus ejes, es necesario establecer la relación que permite expresar un vector definido en un sistema en términos del otro. Este proceso se denomina **rotación entre sistemas de referencia** como se representa en la Figure 1.2.

Sea un sistema $\{A\}$ con ejes \hat{x}_A , \hat{y}_A y \hat{z}_A , y un sistema $\{B\}$ cuyos ejes \hat{x}_B , \hat{y}_B y \hat{z}_B están rotados respecto a $\{A\}$. La posición de un punto \mathbf{r} expresada en el sistema $\{B\}$ se puede convertir al sistema $\{A\}$ mediante una matriz de rotación \mathbf{R} (1.3):

$$\mathbf{r}_A = \mathbf{R} \mathbf{r}_B \quad (1.3)$$

donde \mathbf{R} es una matriz ortogonal de dimensión 3×3 , cuyas columnas están formadas por las componentes de los vectores unitarios de los ejes de $\{B\}$ expresados en el sistema $\{A\}$.

De forma general, la matriz de rotación cumple las siguientes propiedades (Barrientos et al., [1]):

- $\mathbf{R}^T \mathbf{R} = \mathbf{I}$ (ortogonalidad).
- $\det(\mathbf{R}) = +1$ (preserva la orientación).

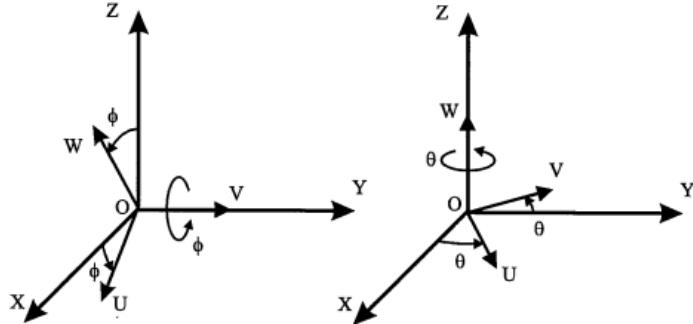


Figura 1.2: Rotación entre sistemas de referencia. De Barrientos et al. [1].

En robótica es habitual trabajar con rotaciones elementales alrededor de los ejes principales. Estas se representan mediante matrices específicas:

- Rotación alrededor del eje x un ángulo θ :

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad (1.4)$$

- Rotación alrededor del eje y un ángulo θ :

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (1.5)$$

- Rotación alrededor del eje z un ángulo θ :

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1.6)$$

Estas rotaciones básicas pueden combinarse para describir cualquier cambio de orientación en el espacio tridimensional. El orden en que se apliquen las rotaciones es determinante, ya que la composición no es conmutativa.

Tal y como señala Barrientos et al. [1], la correcta formulación de estas transformaciones es esencial en la [Cinemática de robots](#), pues permite relacionar las coordenadas de puntos y vectores entre distintos eslabones de un manipulador articulado.

Matrices de transformación homogénea

La combinación de traslaciones y rotaciones es fundamental en la modelización de manipuladores robóticos, ya que permite describir cómo se relacionan entre sí los distintos sistemas de referencia asociados a cada eslabón. Para expresar de manera compacta estas

transformaciones se recurre a las **matrices de transformación homogénea**.

Una matriz de transformación homogénea \mathbf{T} es de dimensión 4×4 y tiene la siguiente forma general:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1.7)$$

donde:

- \mathbf{R} es la matriz de rotación 3×3 que define la orientación de un sistema de referencia $\{B\}$ respecto a otro $\{A\}$.
- $\mathbf{p} = [p_x, p_y, p_z]^T$ es el vector de traslación que representa la posición del origen de $\{B\}$ respecto a $\{A\}$.
- $\mathbf{0}^T = [0 \ 0 \ 0]$ es un vector fila que completa la estructura de la matriz.

Si un punto \mathbf{r}_B se expresa en coordenadas homogéneas en el sistema $\{B\}$, es decir, como $\mathbf{r}_B^h = [x_B, y_B, z_B, 1]^T$, su representación en el sistema $\{A\}$ se obtiene aplicando:

$$\mathbf{r}_A^h = \mathbf{T} \mathbf{r}_B^h \quad (1.8)$$

De esta forma, la matriz homogénea permite, en una sola operación matricial, realizar simultáneamente la rotación y traslación entre dos sistemas de referencia.

Entre las propiedades más relevantes de las transformaciones homogéneas se encuentran (Barrientos et al., [1]):

- La composición de transformaciones se obtiene mediante el producto de matrices homogéneas.
- La inversa de una transformación homogénea también es una matriz homogénea, lo que facilita el cambio entre sistemas de referencia.
- Preservan la estructura geométrica de los puntos y vectores transformados.

En robótica, las matrices de transformación homogénea constituyen la base para la formulación de la [Cinemática](#) directa e inversa, permitiendo describir la posición y orientación del efecto final en función de las variables articulares del manipulador.

Cinemática de robots

La [Cinemática](#) de un [robot](#) estudia el movimiento de su estructura mecánica sin considerar las fuerzas que lo producen. En el caso de manipuladores articulados, el análisis cinemático resulta fundamental para establecer la relación entre las variables articulares

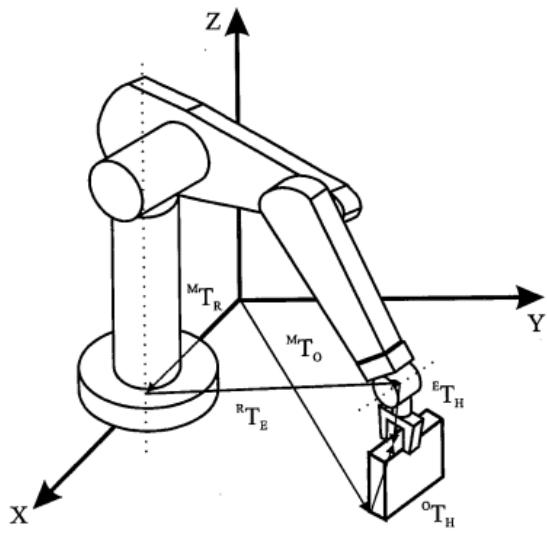


Figura 1.3: Transformación homogénea entre sistemas de referencia. De Barrientos et al. [1].

(ángulos de las articulaciones o desplazamientos lineales) y la posición y orientación del efecto final en el espacio de trabajo.

Se distinguen dos problemas principales en la [Cinemática de robots](#):

- **Cinemática directa:** consiste en determinar la posición y orientación del efecto final a partir de los valores de las variables articulares. Matemáticamente, puede expresarse como una función

$$\mathbf{x} = f(\mathbf{q}) \quad (1.9)$$

donde \mathbf{q} es el vector de coordenadas articulares y \mathbf{x} representa la configuración (posición y orientación) del efecto final en el espacio cartesiano. Este problema siempre tiene una única solución y se resuelve aplicando las matrices de transformación homogénea a lo largo de la cadena [Cinemática del robot](#).

- **Cinemática inversa:** consiste en determinar los valores de las variables articulares necesarios para que el efecto final alcance una posición y orientación deseadas. Se formula como

$$\mathbf{q} = f^{-1}(\mathbf{x}) \quad (1.10)$$

A diferencia de la [Cinemática directa](#), este problema puede tener múltiples soluciones, una solución única o incluso no tener solución, dependiendo de la geometría del [robot](#) y de las restricciones del movimiento.

En robótica, la [Cinemática](#) directa es empleada principalmente para simular y validar el movimiento a partir de valores de entrada conocidos, mientras que la [Cinemática](#) inversa resulta indispensable en tareas de control y planificación de trayectorias, ya que permite

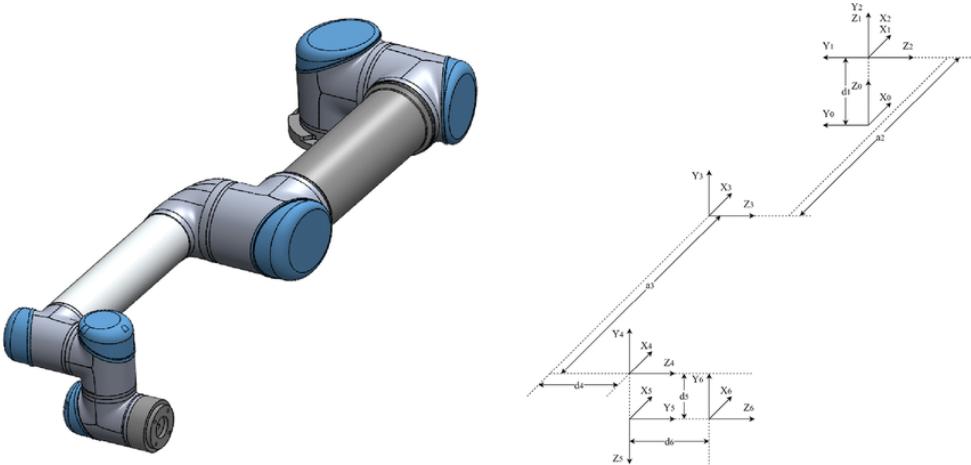


Figura 1.4: Modelo 3D del UR5 y su modelo cinemático correspondiente. De Research Gate et al. [12].

calcular los comandos articulares necesarios para alcanzar un objetivo en el espacio de trabajo.

Tal y como señalan Barrientos et al. [1], el dominio de estos conceptos es esencial en el diseño y control de manipuladores, pues constituye la base de aplicaciones como la interacción humano–robot, la programación por demostración y, en el caso del presente proyecto, la comparación entre el movimiento humano, el ejecutado por el brazo robótico y el registrado por los instrumentos digitalizadores.

En el presente proyecto, la **Cinemática inversa** adquiere una relevancia especial. Dado que los puntos de referencia del movimiento están definidos por las coordenadas de una palabra a reproducir, resulta necesario convertir dichas posiciones cartesianas en valores articulares para el **robot**. De esta forma, para cada punto de la trayectoria se calculan los ángulos de las articulaciones, lo que permite ejecutar el movimiento con mayor precisión y continuidad.

Además, disponer de los valores articulares posibilita el uso de comandos específicos de la librería del **robot** orientados a la interpolación en el espacio articular, lo cual mejora la precisión en la velocidad de ejecución.

Parámetros de Denavit–Hartenberg

Para la modelización sistemática de manipuladores robóticos se emplea de forma habitual la convención de Denavit–Hartenberg (DH), la cual permite describir la geometría de cada eslabón y articulación mediante un conjunto reducido de parámetros. Tal y como señalan Barrientos et al. [1], esta metodología facilita la construcción de las matrices de transformación homogénea entre eslabones consecutivos, simplificando el análisis cinemático.

En esta convención, cada articulación del **robot** se asocia a un sistema de coordenadas, y la relación entre dos sistemas consecutivos $\{i - 1\}$ y $\{i\}$ se describe mediante cuatro

parámetros:

- a_i : distancia entre los ejes Z_{i-1} y Z_i medida a lo largo de X_i .
- α_i : ángulo entre Z_{i-1} y Z_i alrededor del eje X_i .
- d_i : distancia entre los orígenes de los sistemas medida sobre el eje Z_{i-1} .
- θ_i : ángulo de rotación alrededor de Z_{i-1} que lleva a alinear X_{i-1} con X_i .

Con estos parámetros, la matriz de transformación homogénea que relaciona dos eslabones consecutivos se expresa como:

$$T_i^{i-1} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.11)$$

La concatenación de estas transformaciones permite calcular la posición y orientación del efecto final en función de las variables articulares. Este enfoque constituye la base para la [Cinemática](#) directa y, en consecuencia, para la resolución de la [Cinemática](#) inversa.

En el contexto del presente proyecto, la parametrización mediante Denavit– Hartenberg resulta especialmente útil para familiarizarse con la estructura del brazo robótico empleado y para modelar sus movimientos con precisión. De este modo, se obtiene una herramienta sistemática para describir matemáticamente cada articulación y eslabón, lo que facilita la planificación de trayectorias y la validación de resultados experimentales.

1.1.9. Tareas realizadas para cumplir el objeto

En primer lugar, se llevó a cabo una familiarización con el entorno de trabajo, haciendo especial énfasis en el [robot](#), su puesta en marcha, la programación básica y las medidas de seguridad necesarias para su correcta operación.

Posteriormente, se desarrolló un programa en [Python](#) para controlar el [robot Universal Robots 5 \(UR5\)](#) mediante la librería oficial [ur_rtde](#) [11]. El control se realizó utilizando archivos [CSV](#) que contenían las coordenadas de los puntos a dibujar.

Como ejercicio inicial, se escribieron las primeras 100 palabras del libro *Don Quijote de la Mancha* [2], con la tableta gráfica [Wacom Intuos Pro Large \(WIPL\)](#). A continuación, estas mismas palabras se reprodujeron con el [robot](#) sobre la misma tableta, con el fin de comparar los errores de digitalización entre ambos sistemas.

Una vez obtenidos los datos de las palabras escritas por los tres sistemas (humano, robot y tableta), se procedió a su análisis y comparación.

Adicionalmente, se implementó una [Red Neuronal](#) en [Matlab](#) para la predicción de señales a partir de diferentes entradas, considerando tres casos:

- Predicción del movimiento del robot a partir del movimiento humano ([HTR](#)).
- Predicción del movimiento registrado por la tableta a partir del movimiento del robot ([RRT](#)).
- Predicción del movimiento registrado por la tableta a partir del movimiento humano ([HTRT](#)).

y analizando las metricas de error [Root Mean Square Error \(RMSE\)](#), [Mean Absolute Error \(MAE\)](#) y [Mean Squared Error \(MSE\)](#) para cada caso.

De forma complementaria, también se desarrollaron programas en [Matlab](#) para el preprocesamiento de los datos originales, generando un sistema de archivos homogéneo que garantizara un formato común y facilitara la comparación entre conjuntos de datos.

Finalmente, tras el análisis de los resultados obtenidos con la tableta [Wacom Intuos Pro Large \(WIPL\)](#), se repitió el experimento empleando la tableta [Wacom IOT Paper 3 \(IOTP\)](#) y un sensor de movimiento 3D, ampliando así el alcance del estudio.

Ademas, se tuvo que desarrollar e imprimir un soporte para los bolígrafos de las tabletas, así como un soporte para el sensor 3D, que permitiera su correcta sujeción al efecto final del robot.

1.1.10. Normas y referencias

Normativa aplicada

- UNE-EN ISO 10218-1:2012. Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 1: Robots. (ISO 10218-1:2011)
- UNE-EN ISO 10218-2. Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 2: Sistemas robot e integración. (ISO 10218-2:2011).
- UNE-EN ISO 12100:2012. Seguridad de las máquinas. Principios generales para el diseño. Evaluación y reducción del riesgo.
- UNE-EN 60204-1:2019. Seguridad de las máquinas. Equipo eléctrico de las máquinas. Parte 1: Requisitos generales.
- Real Decreto 842/2002, de 2 de agosto, por el que se aprueba el Reglamento electrotécnico para baja tensión.
- Real Decreto 1215/1997, de 18 de julio, por el que se establecen las disposiciones mínimas de seguridad y salud para la utilización por los trabajadores de los equipos de trabajo.
- Ley 31/1995, de 8 de noviembre, de Prevención de Riesgos Laborales.

Referencias bibliográficas

- [1] Antonio Barrientos et al. *Fundamentos de Robótica*. 2.^a ed. McGraw-Hill, 2014.
- [2] Miguel de Cervantes. *El ingenioso hidalgo Don Quijote de la Mancha*. Francisco de Robles, 1605.
- [3] Grup de Tractament del Senyal, Tecnocampus Mataró Maresme. *HandWriting Capture V1.0*. Software para captura de datos con tableta Wacom Intuos Pro Large. 2010.
- [4] JetBrains s.r.o. *PyCharm Community Edition 2021.2.3*. Entorno de desarrollo integrado para Python. 2021. URL: <https://www.jetbrains.com/pycharm/>.
- [5] *Manual de Usuario - UR5*. Consultado el 29 de mayo de 2025. Universal Robots. 2022. URL: <https://www.universal-robots.com/download/manuals/>.
- [6] MathWorks. *MATLAB Documentation*. 2023. URL: <https://www.mathworks.com/help/matlab/> (visitado 22-05-2025).
- [7] *Matplotlib*. <https://matplotlib.org/>. Accedido el 24 de septiembre de 2025.
- [8] *NumPy*. <https://numpy.org/>. Accedido el 24 de septiembre de 2025.
- [9] *Pandas*. <https://pandas.pydata.org/>. Accedido el 24 de septiembre de 2025.
- [10] Python Software Foundation. *Python 3.12.0 Documentation*. Lenguaje de programación interpretado de alto nivel. 2023. URL: <https://docs.python.org/3/>.
- [11] Universal Robots. *UR RTDE: Real-Time Data Exchange Interface*. Documentación técnica oficial. 2022. URL: <https://www.universal-robots.com> (visitado 22-05-2025).
- [12] *Universal Robot 5 (UR5): 3D model and corresponding kinematic model — Figure on ResearchGate*. Figura consultada en línea. ResearchGate. URL: https://www.researchgate.net/figure/Universal-Robot-5-UR5-left-3-D-model-right-Corresponding-kinematic-model_fig1_355352649 (visitado 15-09-2025).
- [13] Ltd. Wacom Co. *Manual de usuario de la Wacom Intuos Pro*. Versión PDF consultada online. 2021. URL: <https://www.wacom.com> (visitado 22-05-2025).
- [14] Wacom Co., Ltd. *How to use IoT Paper*. Versión de investigación. Documento de soporte para la tableta IoT Paper. Nov. de 2023.

1.2. Descripción del trabajo realizado

1.2.1. Visión general de la solución adoptada

La solución adoptada se fundamenta en el desarrollo de una plataforma experimental que integra distintos dispositivos de captura de escritura y un robot colaborativo **UR5**, con

el objetivo de analizar y modelar los errores introducidos en el proceso de digitalización y reproducción de trazos manuscritos.

El sistema completo se organizó en cuatro fases principales:

1. **Captura de datos.** Se emplearon tres instrumentos distintos: la tableta [Wacom Intuos Pro Large \(WIPL\)](#), la tableta prototipo [Wacom IOT Paper 3 \(IOTP\)](#) y un sistema captor 3D construido a partir de potenciómetros y Arduino. Cada dispositivo registró las trayectorias manuscritas en sus propios formatos de salida, posteriormente unificados.
2. **Control del robot.** A través de [Python](#) y la librería [UR_RTDE](#) se estableció comunicación en tiempo real con el [UR5](#). El robot recibió como entrada las trayectorias previamente capturadas y las reprodujo sobre el instrumento mediante un útil adaptado al extremo de su efecto final.
3. **Procesamiento y análisis.** En [Matlab](#) se implementaron rutinas de calibración y transformación de coordenadas, asegurando la compatibilidad de datos entre los diferentes instrumentos. Además, se aplicaron técnicas de preprocesado para reducir ruido, homogeneizar longitudes de señales y calcular métricas de error entre las trayectorias humanas y robóticas.
4. **Modelado del error.** Se entrenaron redes neuronales con el fin de predecir la señal generada realmente a partir de la referencia humana. Este enfoque permitió evaluar cuantitativamente el error introducido por cada dispositivo y por el propio robot, sin necesidad de ejecutar físicamente todas las pruebas.

Gracias a esta estructura modular, la plataforma permitió comparar el comportamiento de distintos dispositivos de captura bajo un mismo flujo de trabajo, cuantificando los errores introducidos en cada etapa y extrayendo conclusiones sobre la fiabilidad y aplicabilidad de los distintos métodos de digitalización.

1.2.2. Diagrama de flujo del proyecto

El diagrama de flujo mostrado en la Figura 1.5 resume de manera esquemática el funcionamiento global del sistema desarrollado en el TFG. Se distinguen claramente tres bloques principales que se corresponden con las fases de captura de datos, procesado y análisis, y reproducción de la escritura mediante el robot colaborativo.

En la primera fase, los distintos instrumentos digitalizadores ([WIPL](#), [IOTP](#) y sistema captor 3D) registran las trazas manuscritas generadas por el usuario. Los datos se almacenan en ficheros individuales por palabra, estructurados en formato [CSV](#), garantizando una organización homogénea y compatible con los entornos de procesado posteriores.

La segunda fase corresponde al procesado de la información. A través de scripts desarrollados en MATLAB y Python se lleva a cabo la estandarización de los ficheros y la calibración de señales.

Finalmente, en la tercera fase se realiza la comunicación con el robot **UR5** mediante la librería **UR_RTDE**. Las trayectorias capturadas se transmiten al robot, que se encarga de reproducirlas directamente sobre el mismo dispositivo utilizado en la captura inicial. De este modo, es posible comparar de manera directa la escritura humana y la réplica robótica bajo un mismo flujo de trabajo.

En conjunto, el diagrama de flujo ilustra la naturaleza modular y secuencial de la solución adoptada, donde cada bloque cumple una función específica pero integrada en un pipeline común que asegura la trazabilidad completa de los datos desde su captura hasta el modelado del error.

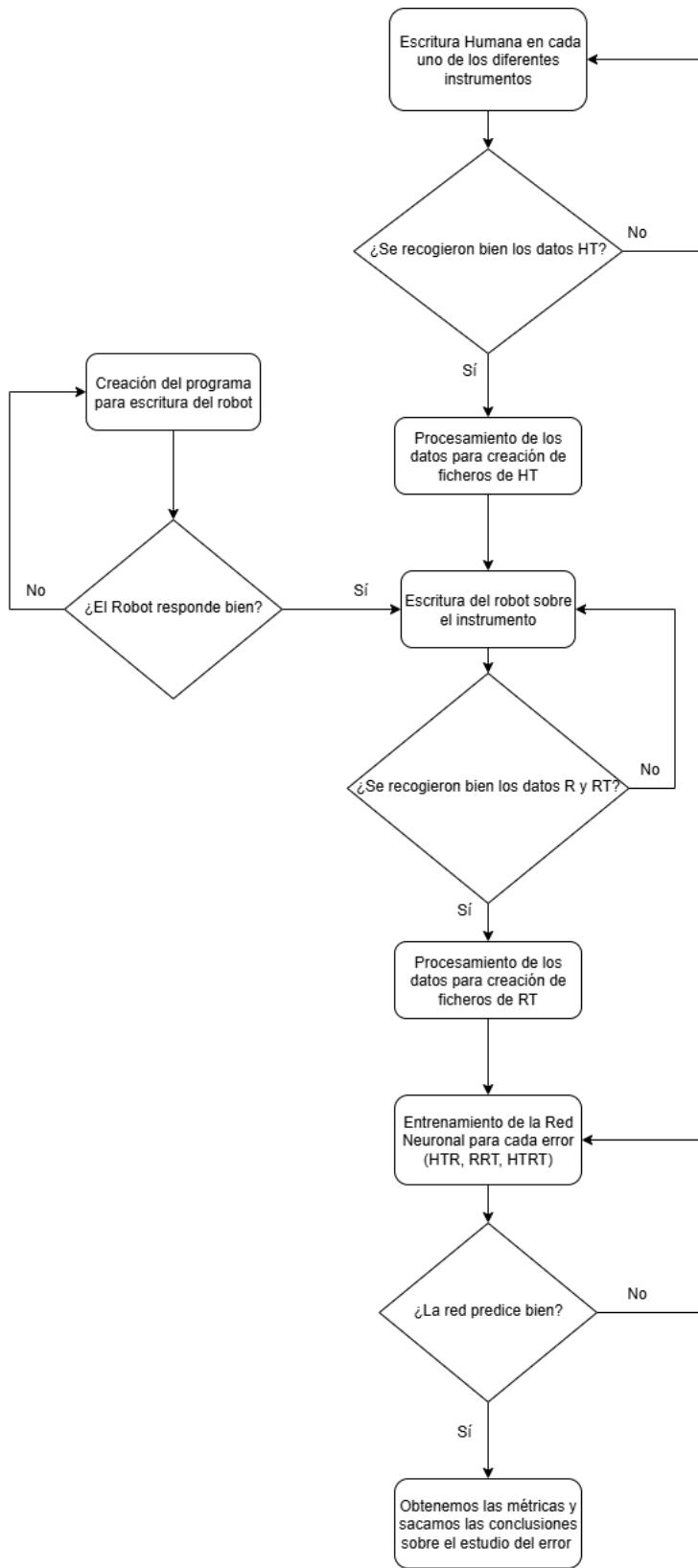


Figura 1.5: Diagrama de flujo del proyecto.

1.2.3. Requisitos de diseño

Durante la fase inicial del proyecto se establecieron una serie de requisitos de diseño que guiaron el desarrollo de la solución adoptada. Estos requisitos se clasifican en funcionales y no funcionales.

Requisitos funcionales

- El sistema debe permitir la captura de escritura manuscrita mediante distintos dispositivos de entrada (Wacom [WIPL](#), Wacom [IOTP](#) y sistema captor 3D).
- Los datos obtenidos deben almacenarse en ficheros estructurados en formato CSV, con un fichero por palabra o ejercicio. Cada fichero debe contener cuatro columnas: Tiempo (s), Coordenada X, Coordenada Y y Coordenada Z.
- La base de datos debe organizarse en una estructura jerárquica que distinga el dispositivo de captura empleado ([WIPL](#), [IOTP](#), P) y el origen de la señal (HT, R, RT).
- El robot colaborativo [UR5](#) debe ser capaz de reproducir las trayectorias capturadas escribiendo directamente sobre el mismo instrumento utilizado en la captura de datos.
- El software de control debe gestionar la comunicación en tiempo real con el robot, garantizando la transmisión continua de trayectorias.
- El sistema debe incluir un modelo predictivo, basado en redes neuronales, cuyo entrenamiento permita obtener métricas de error ([RMSE](#), [MAE](#), [MSE](#)) para comparar la señal real con la predicha y evaluar así la precisión del modelo.

Requisitos no funcionales

- El sistema debe ser compatible con los formatos de datos generados por los dispositivos empleados (CSV).
- El software desarrollado debe ejecutarse en entornos multiplataforma (Python, MATLAB) y ser fácilmente adaptable a otros escenarios de prueba.
- El diseño debe respetar los principios de seguridad aplicables a robots colaborativos, siguiendo las normas ISO 10218 e ISO/TS 15066.
- La solución debe permitir la trazabilidad completa de los datos, desde la captura inicial hasta la reproducción robótica y el análisis posterior.

- El sistema debe mantenerse modular, de forma que sea posible incorporar nuevos dispositivos de captura o algoritmos de análisis sin necesidad de rediseñar la arquitectura completa.

1.2.4. Hardware y software utilizados

Hardware

Brazo robótico Universal Robots 5 (UR5) de Universal Robots. El UR5 es un robot colaborativo de seis grados de libertad desarrollado por Universal Robots, diseñado para tareas de manipulación ligera y programación sencilla [5]. En este proyecto se utilizó como actuador principal encargado de reproducir sobre la tableta los mismos trazos previamente capturados de la escritura humana, garantizando la repetibilidad y precisión necesarias para el análisis de errores de digitalización.

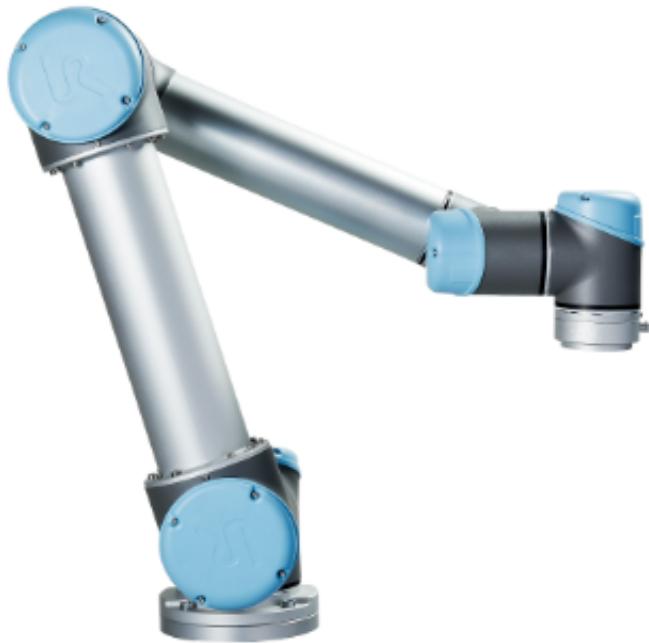


Figura 1.6: Brazo robótico UR5 de Universal Robots de [5].

Tableta gráfica Wacom Intuos Pro Large (WIPL) de Wacom. La Wacom Intuos Pro Large (WIPL) es una tableta digitalizadora profesional desarrollada por Wacom. Según el manual de usuario [13], este dispositivo admite entrada tanto con lápiz como táctil, dispone de un área activa amplia (311×216 mm en el modelo Large) y está equipada con ocho teclas programables *ExpressKeys* y un *Touch Ring* multifunción que facilitan la interacción con aplicaciones gráficas [13]. El lápiz inalámbrico que la acompaña no requiere batería, es sensible a la presión y a la inclinación, lo que permite registrar con precisión la dinámica del trazo humano.

En el presente proyecto, la [WIPL](#) se empleó como dispositivo de referencia para la captura de escritura manuscrita. Los datos obtenidos se almacenaron en archivos [CSV](#) que sirvieron posteriormente como entrada para el robot [UR5](#) y para los análisis comparativos. Gracias a su resolución y estabilidad, la tableta proporcionó una base fiable para el estudio de errores de digitalización y para el entrenamiento de la red neuronal.



Figura 1.7: Tableta gráfica Wacom Intuos Pro Large de [13].

Tableta gráfica Wacom IOT Paper 3 (IOTP) de Wacom. La [Wacom IOT Paper 3 \(IOTP\)](#) es una tableta digitalizadora en fase de prototipo desarrollada por Wacom, concebida para investigación académica y aplicaciones educativas. El modelo utilizado en este proyecto corresponde a la versión de 10,3 pulgadas, equipada con una pantalla de tinta electrónica (*E-Ink Digital Ink Screen Display*) de 297×182 mm, con resolución de 1872×1404 píxeles (226 dpi) [14].

La tableta incorpora un lápiz digital *Wacom EMR*, sensible a la presión, a la altura y a la inclinación, lo que permite capturar de manera precisa las características dinámicas de la escritura manuscrita. Los datos se registran en formato *InkML*, que incluye coordenadas (x, y) , presión ejercida, inclinación del lápiz y marcas temporales. Estos ficheros se exportan directamente a un ordenador mediante conexión USB Type-C, siendo reconocida por el sistema como un dispositivo de almacenamiento masivo. El software proporcionado por la propia empresa genera tres tipos de archivos asociados a cada captura: un mapa de bits ([.bmp](#)), un fichero tabulado ([.csv](#)) y un fichero estructurado en *InkML*. Entre ellos, el formato [.csv](#) fue el seleccionado para la integración con el robot, ya que permitía transferir de manera directa las coordenadas a reproducir en el experimento [14].

En el presente proyecto, la [IOTP](#) se empleó como dispositivo alternativo de captura de escritura, con el objetivo de estudiar el error asociado al uso de distintos instrumentos de digitalización. Cada tableta, al contar con características técnicas y niveles de precisión diferentes, ofrece resultados propios que influyen en el análisis final. Los datos obtenidos con la [IOTP](#) se procesaron siguiendo el mismo flujo de trabajo que en el caso de la [WIPL](#),

lo que permitió evaluar cómo las diferencias entre dispositivos se reflejan en los errores de digitalización y en las posibles aplicaciones futuras de este tipo de sistemas.



Figura 1.8: Tableta gráfica Wacom IOTP de [14].

Sensor de movimiento 3D mediante potenciómetros de 3 ejes. Además de las tabletas gráficas, se utilizó un sistema de captura de movimiento tridimensional diseñado de forma experimental. Este dispositivo estaba compuesto por tres potenciómetros conectados a una placa *Arduino*, cuya señal permitía estimar la posición de un punto en el espacio tras un proceso de calibración previo.

Para la adquisición de datos se desarrolló un programa en **Matlab**, que se comunicaba directamente con el entorno *Arduino IDE* mediante el puerto serie. El código en la placa *Arduino* se encargaba de leer de forma periódica las entradas analógicas asociadas a los potenciómetros y transmitirlas a través del puerto serie a alta velocidad. Matlab recibía estas tramas de datos en tiempo real, ejecutaba la calibración y procesaba las señales para reconstruir la trayectoria espacial realizada.

Como resultado, el sistema generaba ficheros en formato **CSV** que contenían las coordenadas tridimensionales (x, y, z) de cada punto registrado. Estos datos se integraron en el mismo flujo de trabajo que los obtenidos con las tabletas gráficas, lo que permitió analizar el error de digitalización no solo en el plano, sino también en el eje de profundidad.



Figura 1.9: Sensor de movimiento 3D mediante potenciómetros y robot.

Ordenador de sobremesa [TODO: Incluir una tabla con las características técnicas del ordenador.]

Impresora 3D [TODO: Modelo y características técnicas de la impresora 3D.]

Software

En el desarrollo del proyecto se emplearon distintos programas y entornos de programación. Cada uno de ellos cumplió una función específica tanto en la fase de captura de datos como en la de procesado y validación de resultados.

Entorno de desarrollo IDE PyCharm Community Edition 2021.2.3. *PyCharm* es un entorno de desarrollo integrado ([IDE](#)) especializado en programación con Python. Ofrece herramientas avanzadas para la edición de código, depuración, control de versiones y gestión de librerías, lo que facilita el desarrollo estructurado de proyectos de software. La versión *Community Edition* es gratuita y de código abierto, manteniendo la mayoría de las funcionalidades necesarias para el ámbito académico.[\[4\]](#)

En el contexto del TFG se empleó como entorno principal para la implementación de los programas en Python. PyCharm permitió integrar de manera sencilla las librerías externas utilizadas, organizar los módulos de código y depurar los scripts encargados del procesado de datos y de la comunicación con el robot [UR5](#).

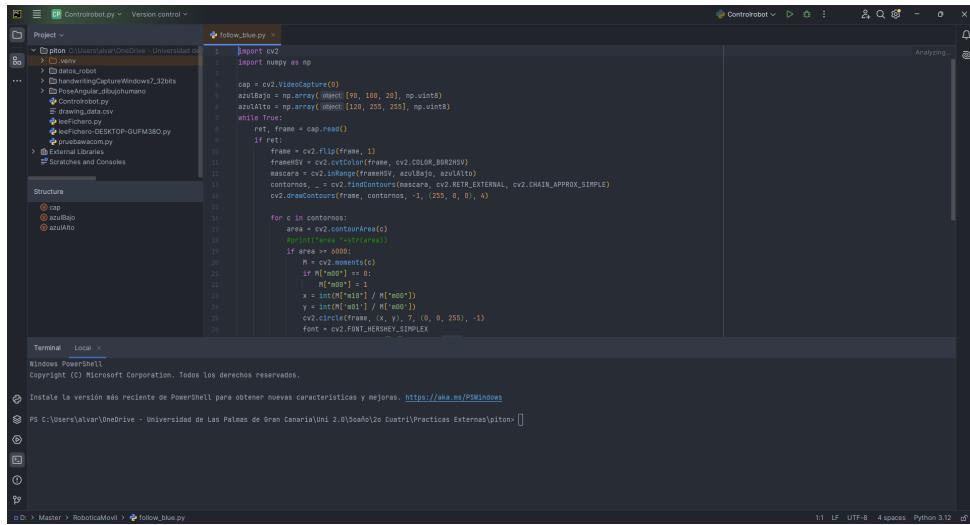


Figura 1.10: Entorno de desarrollo PyCharm Community Edition 2021.2.3.

Lenguaje de programación Python 3.12.0. *Python* es un lenguaje de programación interpretado, de alto nivel y con una sintaxis sencilla, ampliamente extendido en ámbitos científicos y de ingeniería. Su diseño orientado a la legibilidad y la gran disponibilidad de librerías lo convierten en una herramienta idónea para el desarrollo de aplicaciones de análisis de datos, inteligencia artificial y control de hardware. [10]

En el marco del TFG se empleó la versión 3.12.0, que permitió implementar los módulos encargados de la lectura y procesado de los ficheros CSV generados por los dispositivos de captura. Asimismo, Python se utilizó como interfaz de comunicación con el robot UR5 mediante la librería UR_RTDE, transmitiendo las trayectorias manuscritas al manipulador para su reproducción física.

Librerías de Python

- **NumPy:** empleada para cálculos numéricos, manipulación de matrices y operaciones algebraicas necesarias en el tratamiento de señales [8].
- **Pandas:** permitió organizar los datos en estructuras tipo tabla (DataFrames), facilitando la lectura y escritura de archivos CSV [9].
- **Matplotlib:** utilizada para la creación de gráficos y la representación visual de los resultados obtenidos en las pruebas [7].
- **UR_RTDE:** proporcionó la interfaz de comunicación en tiempo real con el robot UR5, haciendo posible el envío de trayectorias y la monitorización de variables del manipulador [11].

Entorno de desarrollo MATLAB R2023b *MATLAB* es un entorno de programación y cálculo numérico ampliamente utilizado en investigación y desarrollo científico– técnico.

Proporciona un lenguaje propio orientado a matrices, así como un extenso conjunto de librerías para el análisis de datos, representación gráfica, modelado matemático y diseño de algoritmos.[6]

En el presente TFG se empleó la versión R2023b, que permitió implementar rutinas de preprocessado y calibración de los datos capturados por los diferentes instrumentos de escritura. Asimismo, MATLAB se utilizó para realizar análisis estadísticos de error y para el entrenamiento y validación de la red neuronal destinada a evaluar el rendimiento de cada dispositivo. Adicionalmente, para el caso 3D, el entorno sirvió de puente con la placa Arduino, mediante la adquisición en tiempo real de los valores analógicos de los potenciómetros, lo que facilitó la integración de los datos en el flujo de trabajo general del proyecto.

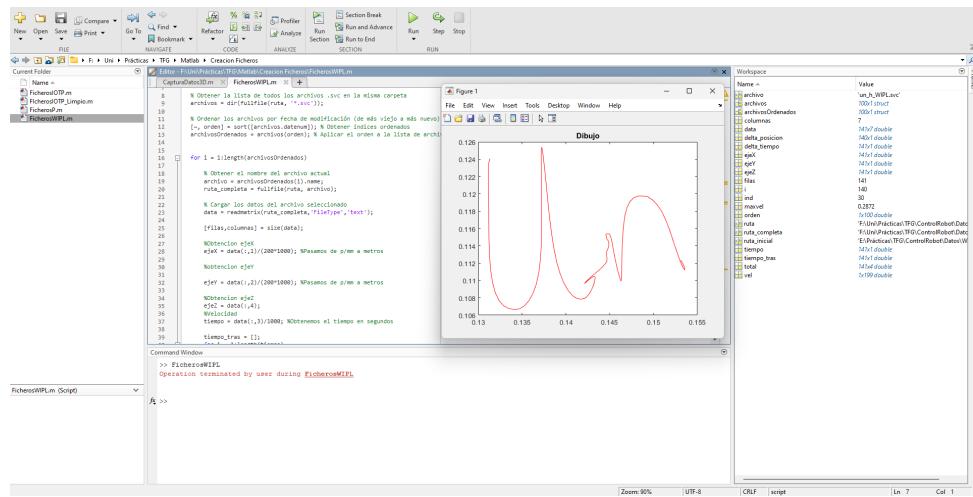


Figura 1.11: Entorno de desarrollo MATLAB R2023b.

Software para captura de WIPL. HandWriting Capture. Para la adquisición de datos con la tableta Wacom WIPL se empleó el software *HandWriting Capture V1.0*, desarrollado por el Grup de Tractament del Senyal del Tecnocampus Mataró Maresme.[3]

Esta aplicación permitió registrar de manera directa las trazas manuscritas realizadas sobre la tableta y exportarlas en archivos CSV, lo que facilitó su posterior integración en el flujo de trabajo del proyecto. El programa ofrece una interfaz sencilla en la que es posible crear o cargar usuarios, ejecutar ejercicios de escritura y almacenar los resultados en ficheros de datos. Dichos archivos se guardan automáticamente en la carpeta data de la aplicación, garantizando una organización homogénea para todas las sesiones de captura.[3]

En el marco del TFG, esta herramienta se utilizó como fuente principal de datos manuscritos para la tableta WIPL, asegurando la compatibilidad con los entornos de procesado implementados en Python y MATLAB.

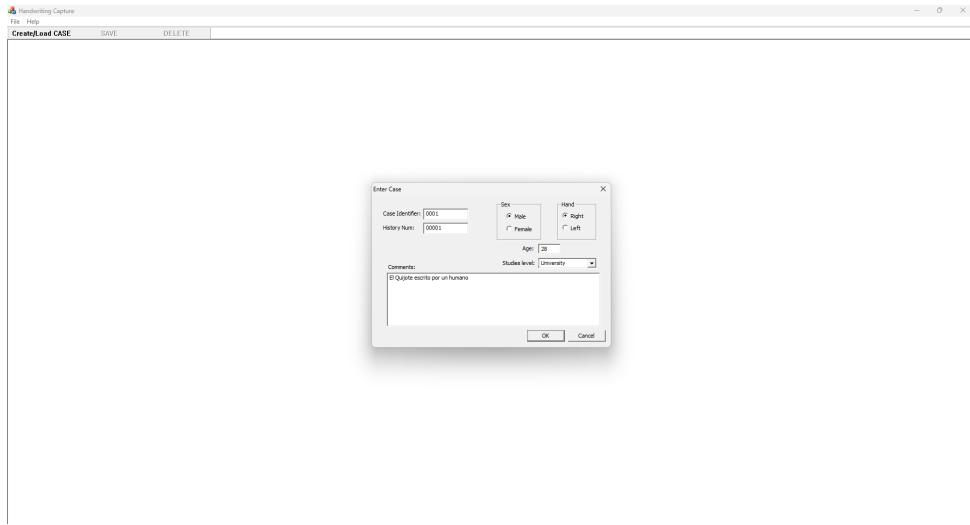


Figura 1.12: Software de captura de datos desde la tableta Wacom [WIPL](#).

Software para captura de IOTP La captura de datos de la tableta IOTP se realizó a través del software experimental desarrollado por Wacom y distribuido junto con el dispositivo. El programa está basado en el protocolo MTP (*Media Transfer Protocol*), lo que permite que la tableta se comporte como un dispositivo de almacenamiento externo, similar a una cámara digital o a un teléfono móvil. De este modo, los archivos de escritura pueden transferirse directamente al ordenador sin necesidad de aplicaciones adicionales[14].

El software genera y gestiona ficheros en formato InkML, que contienen tanto las coordenadas (x, y) como la información dinámica asociada (presión, inclinación y marcas temporales). Además, junto a cada captura se producen archivos complementarios en formato .bmp y .csv, siendo este último el seleccionado en el marco del TFG por su compatibilidad con los entornos de procesado en Python y MATLAB [14].

La aplicación proporciona también herramientas auxiliares, como el programa *Inkml-Converter*, que permite generar ficheros InkML a partir de imágenes de plantilla en mapa de bits, para ser utilizadas como fondo de referencia en la escritura. Gracias a esta funcionalidad, es posible personalizar los ejercicios de captura y mantener un control preciso sobre el contenido manuscrito.

En el contexto del TFG, este software se empleó para registrar, almacenar y transferir al ordenador las sesiones de escritura realizadas con la IOTP, integrando posteriormente los datos en el flujo de trabajo común junto con los obtenidos de la WIPL.

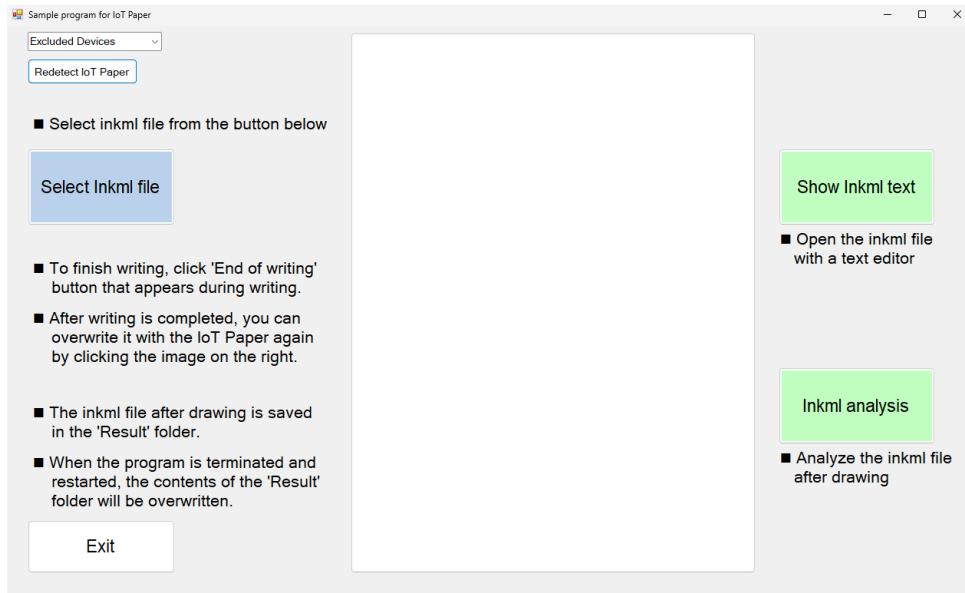


Figura 1.13: Software de captura de datos desde la tableta Wacom IOTP.

Software Arduino IDE. El *Arduino IDE* es el entorno de desarrollo oficial para la programación de placas basadas en microcontroladores de la plataforma Arduino. Este software permite escribir, compilar y cargar programas (denominados *sketches*) en la placa a través de una interfaz sencilla y multiplataforma. Además, incluye un monitor serie que facilita la comunicación en tiempo real entre el microcontrolador y el ordenador.

En el marco del TFG, el Arduino IDE se utilizó para desarrollar y transferir el código encargado de la lectura analógica de los tres potenciómetros del sistema captor 3D. Los valores obtenidos se enviaban de forma continua por el puerto serie, de modo que pudieran ser adquiridos posteriormente en MATLAB para su procesado, calibración y análisis.

Figura 1.14: Entorno de desarrollo Arduino IDE.

1.2.5. Dificultades encontradas y soluciones adoptadas

Durante el desarrollo del proyecto se identificaron diferentes dificultades relacionadas con la captura y el análisis de datos. A continuación, se describen las más relevantes junto con las medidas adoptadas para su resolución, incorporando ejemplos gráficos que ilustran el efecto de cada problema.

Errores por separación excesiva del lápiz

En la captura realizada con la tableta WIPL se observó que, cuando el lápiz se separaba demasiado de la superficie, la tableta dejaba de registrar puntos intermedios. Esto provocaba saltos bruscos en las coordenadas registradas y picos anómalos en la señal de velocidad, que no correspondían a un movimiento real de escritura. Esto se traducía en movimientos bruscos del robot pasando de una coordenada a otra sin suavizado, lo que afectaba a la calidad de la reproducción. La solución adoptada consistió en mantener el lápiz siempre próximo a la superficie durante el ejercicio, incluso en los movimientos de transición sin contacto directo, evitando así pérdidas de muestreo.

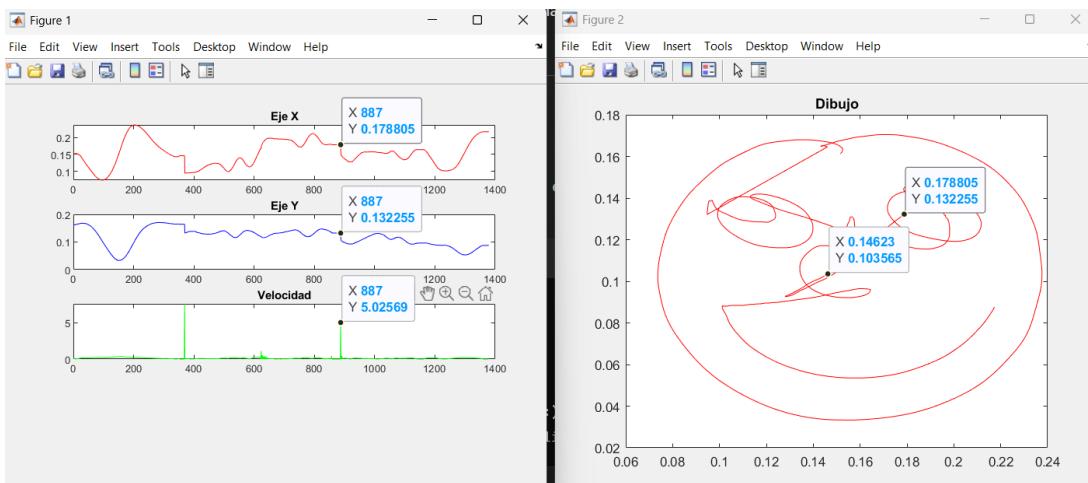


Figura 1.15: Ejemplo de pico de velocidad generado por ausencia de puntos intermedios al separar demasiado el lápiz de la tableta.

Errores por activación involuntaria de botones

Otro problema detectado estuvo asociado al lápiz digital de la WIPL. Durante la escritura, la pulsación accidental de los botones laterales provocaba la aparición de valores anómalos en la coordenada Z (valores distintos de 0 o 1), lo que se traducía en distorsiones en la señal y nuevos picos de velocidad.

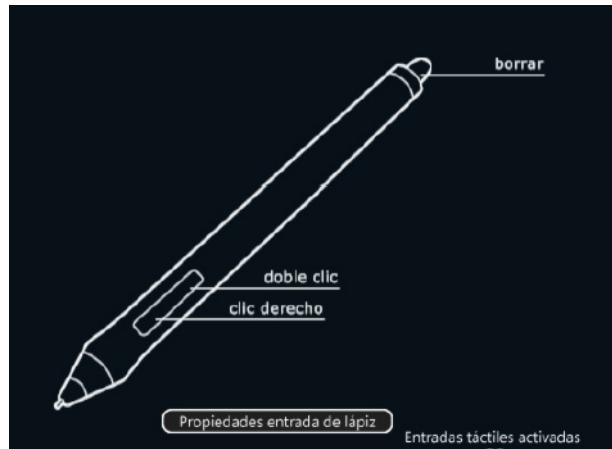


Figura 1.16: Lápiz digital de la tableta Wacom [WIPL](#), con los botones laterales que pueden activarse involuntariamente durante la escritura.

La solución consistió en evitar la activación de los botones durante las sesiones experimentales y, en caso necesario, deshabilitar sus funciones en la configuración del dispositivo.

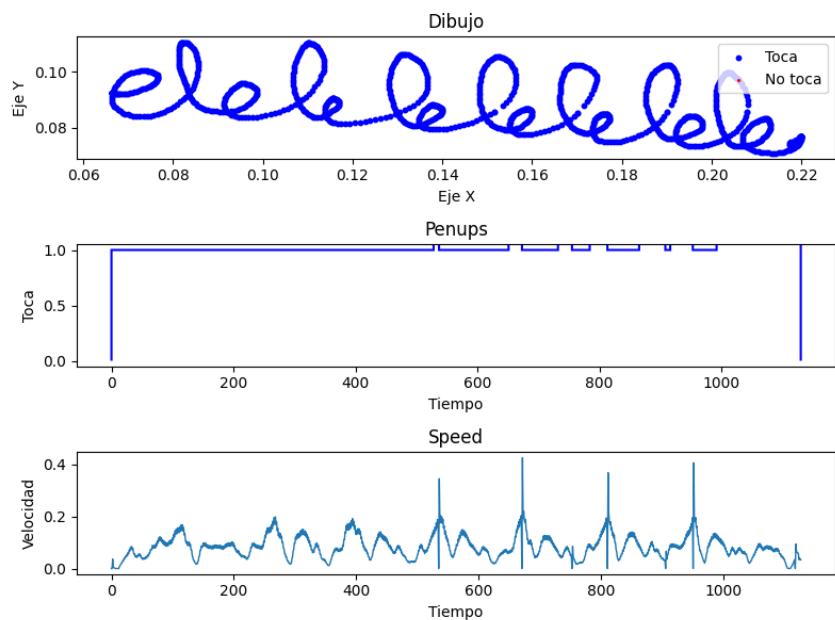


Figura 1.17: Valores anómalos de la coordenada Z asociados a la activación involuntaria de los botones del lápiz, que generan distorsiones en la señal de velocidad.

Desfases entre señales humanas y robóticas

Durante la comparación de trayectorias [Robot](#) y [Robot-Tableta](#) se identificaron desfases tanto espaciales como temporales. Este problema dificultaba la superposición directa de señales, lo que impedía un análisis cuantitativo inmediato.

Para abordar esta incidencia se implementaron técnicas de preprocessado en MATLAB. En particular, se aplicó el centrado de trayectorias mediante el cálculo del centroide, con el fin de reducir desplazamientos espaciales iniciales, y se utilizó interpolación temporal para homogeneizar la longitud de las señales. Ambas correcciones se mantuvieron dentro del

flujo de trabajo, dado que no modificaban sustancialmente el error real de los dispositivos y mejoraban la comparabilidad de las trayectorias.

En cambio, la técnica de alineación temporal *Dynamic Time Warping* (DTW), aunque inicialmente considerada, fue descartada. Su aplicación implicaba alterar la dinámica original de las señales, lo que hubiera supuesto perturbar los errores reales que se pretendían analizar. Por este motivo, se decidió no emplearla y trasladar el tratamiento de los desfases restantes al modelado con redes neuronales, permitiendo que el modelo aprendiera de manera natural las discrepancias presentes entre trayectorias.

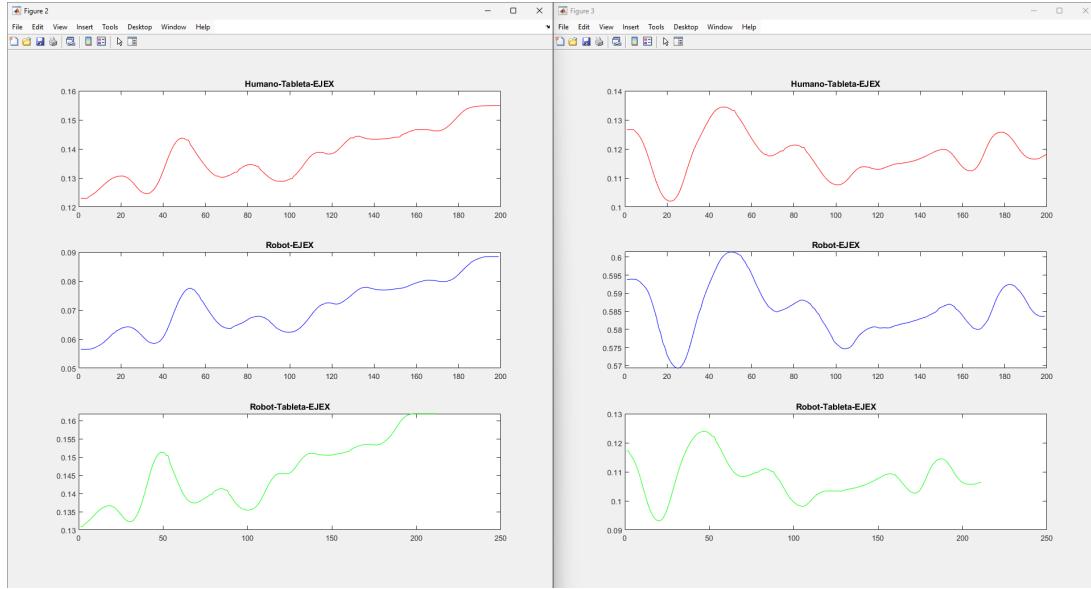


Figura 1.18: Ejemplo de desfase entre la trayectoria **R** y la trayectoria **RT**.

Velocidad excesiva en la captura 3D

En el caso del sistema captor 3D, la primera toma de datos realizada por el usuario se llevó a cabo con una velocidad de ejecución demasiado elevada. Al reproducirse en el robot, esta dinámica suponía un riesgo para la integridad del sistema de sujeción y del propio sistema captor 3D, ya que los movimientos eran más rápidos de lo que el montaje podía tolerar.

Ante este problema se decidió repetir la toma de datos, esta vez con una escritura más lenta y precisa por parte del usuario. De este modo, el robot pudo reproducir las trayectorias sin poner en riesgo los elementos mecánicos ni comprometer la estabilidad del experimento.

En conjunto, la resolución de estas incidencias permitió obtener un flujo de datos más robusto y representativo, garantizando que los errores analizados fuesen intrínsecos al sistema y no fruto de artefactos de captura o preprocessado.

1.2.6. Disposición física de los elementos

La Figura 1.19 muestra la disposición física y las conexiones entre los diferentes componentes del sistema experimental. El ordenador actúa como unidad central de procesamiento, ejecutando los programas desarrollados en Python y MATLAB para la lectura de datos, el preprocesado de ficheros y el envío de trayectorias al robot.

El ordenador se conecta al controlador del robot UR5 a través de un cable Ethernet, lo que permite la comunicación en tiempo real mediante la librería UR_RTDE. De manera paralela, el instrumento digitalizador empleado en cada experimento ([WIPL](#), [IOTP](#) o sistema captor 3D) se conecta al ordenador por USB, asegurando la adquisición directa de los datos manuscritos en formato digital.

El controlador del robot está vinculado físicamente al *teach pendant* (interfaz manual), que permite ejecutar operaciones básicas de configuración y seguridad. Finalmente, el robot recibe las trayectorias procesadas desde el ordenador y las reproduce directamente sobre el instrumento digitalizador, garantizando así la coherencia entre la escritura humana y la réplica robótica.

Esta disposición modular asegura la trazabilidad de los datos, desde la captura inicial hasta la reproducción física de las trayectorias, y facilita la sustitución de instrumentos de captura sin necesidad de modificar la arquitectura general del sistema.

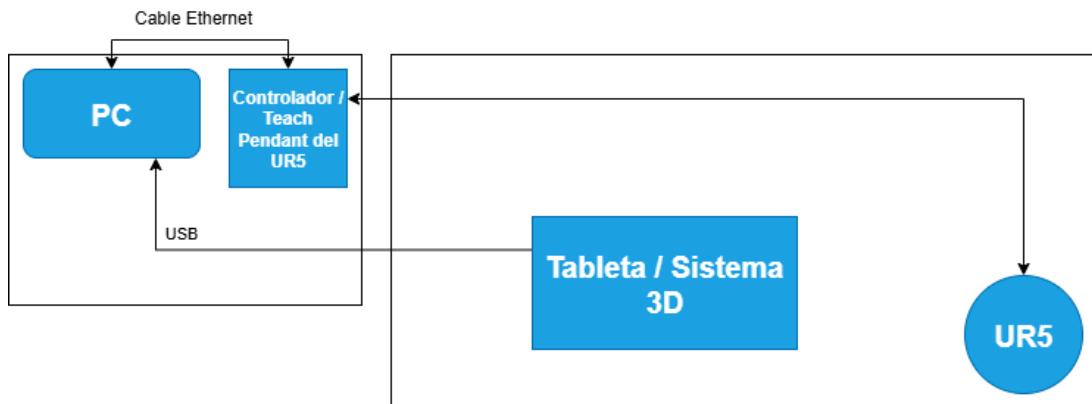


Figura 1.19: Esquema de la disposición física del sistema.

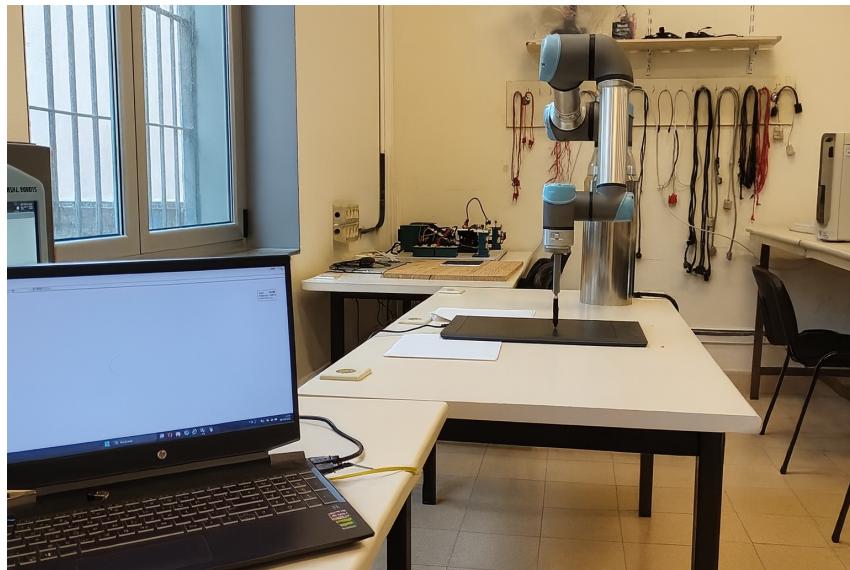


Figura 1.20: Disposición real del sistema en el laboratorio.

1.2.7. Descripción detallada de la solución

[TODO: TODO lo gordo] [TODO: Sacar foto de la configuracion manual de las herramientas del robot en el teach pendant] [TODO: Poner todo esto en la parte de 3D]

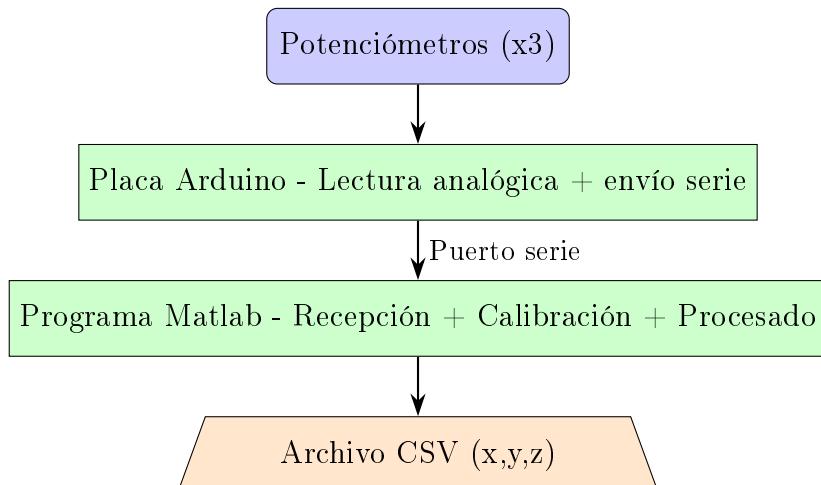


Figura 1.21: Flujo de funcionamiento del sistema captor 3D (vertical).

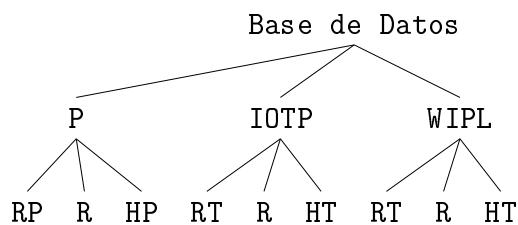


Figura 1.22: Estructura jerárquica de los directorios de la base de datos

Tiempo (s)	Coordenada X	Coordenada Y	Coordenada Z
0.000	12.345	67.890	0.000
0.008	12.350	67.895	0.000
0.016	12.360	67.910	1.000
:	:	:	:

Cuadro 1.1: Estructura de los ficheros de datos por palabra.

1.2.8. Resultados

Metodología experimental

(Contenido pendiente.)

Aplicación del modelo entrenado

(Contenido pendiente.)

Resultados obtenidos en metricas clave

1.2.9. Conclusiones finales

(Contenido pendiente.)

1.2.10. Posibles mejoras y trabajo futuro

(Contenido pendiente.)

Capítulo 2

Documento II: Memoria de Cálculo

- 2.1. Hipótesis, condiciones de contorno y criterios de diseño
- 2.2. Modelos matemáticos y ecuaciones empleadas
- 2.3. Procedimientos de cálculo y dimensionamiento
- 2.4. Resultados de cálculo (tablas y figuras)
- 2.5. Verificación, validación y análisis de incertidumbre
- 2.6. Conclusiones de la memoria de cálculo

Capítulo 3

Documento III: Pliego de Condiciones Técnicas

- 3.1. Condiciones generales
- 3.2. Clasificación de la instalación
- 3.3. Criterios de diseño y ejecución
- 3.4. Materiales y equipos
- 3.5. Condiciones de montaje
- 3.6. Programa de mantenimiento
- 3.7. Pruebas y puesta en servicio
- 3.8. Documentación necesaria

Capítulo 4

Documento IV: Presupuesto

4.1. Cuadro de precios unitarios

4.2. Medición y valoración

4.3. Resumen por capítulos

4.4. Presupuesto de ejecución material

4.5. Costes indirectos y gastos generales

4.6. Presupuesto total

Capítulo 5

Documento V: Esquemas y Conexionados

5.1. Diagrama de bloques del sistema

5.2. Esquemas de comunicaciones / red

Capítulo 6

Documento VI: Anexos

6.1. Listados de código y scripts

[TODO: Código fuente del control del robot en Python.] [TODO: Código fuente del preprocesamiento de datos en Matlab para cada caso.] [TODO: Código de Arduino IDE para el sensor 3D.] [TODO: Código fuente de la red neuronal en Matlab.]

6.2. Manuales / datasheets relevantes