UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE VISUALIZACIÓN DE DATOS EN TIEMPO REAL DE UN ENTRENADOR LAPAROSCÓPICO

TERESA GÓMEZ FERNÁNDEZ

2016

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

Reunido el tribunal examinador en el día de la fecha, constituido por

Presidente:	Dr D.	Carlos González Bris
Vocal:	Dr D.	José Jesús Fraile Ardanuy
Secretaria:	Dra D^a	Patricia Sánchez González
Suplente:	Dr D.	Alfredo Sanz Hervás

para juzgar el Trabajo Fin de Grado titulado:

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE VISUALIZACIÓN DE DATOS EN TIEMPO REAL DE UN ENTRENADOR LAPAROSCÓPICO

de la alumna D^a. Teresa Gómez Fernández
dirigido por D. Álvaro Gutiérrez Martín

Acuerdan otorgar la calificación de: _____

Y, para que conste, se extiende firmada por los componentes del tribunal, la presente diligencia

Madrid, a _____ de _____ de _____

El Presidente | El Vocal | El Secretario

Fdo: _____ Fdo: _____ Fdo: _____

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE VISUALIZACIÓN DE DATOS EN TIEMPO REAL DE UN ENTRENADOR LAPAROSCÓPICO

TERESA GÓMEZ FERNÁNDEZ

2016

Resumen

El principal objetivo del Trabajo Fin de Grado consiste en implementar un sistema de visualización en tiempo real de los datos procedentes de un entrenador laparoscópico. La caja entrenadora de la que se dispone consta de dos instrumentos tubulares (con asideros a modo de pinzas laparoscópicas) y ocho encoders en cuadratura que registran sus movimientos.

Las ocho señales de los encoders se procesan en una tarjeta de desarrollo (Arduino DUE), previo paso por un circuito de adaptación que estructura el entramado de cables. El Arduino DUE tiene la función de enviar por el puerto serie al ordenador los datos de posición relativa con el formato especificado.

En el ordenador se decodifican los datos recibidos para ser mostrados en el programa de visualización elegido. Para ello, se han desarrollado dos programas distintos: uno orientado a la representación de gráficas bidimensionales y otro a la representación tridimensional de los instrumentos en cuestión.

Para la coordinación de los distintos elementos y el desarrollo del sistema global ha sido imprescindible tener en cuenta las limitaciones de la tarjeta de desarrollo utilizada, las especificaciones de tiempo real previstas y las necesidades futuras de realimentación háptica.

Palabras clave

Simulador laparoscópico, Encoder, Arduino DUE, Visualización.

Agradecimientos

A mi familia, por el apoyo que me han dado siempre para lograr mis metas personales.

A todo el personal y estudiantes de Robolabo, especialmente a mi tutor Álvaro Gutiérrez, por su paciencia y la ayuda que me han prestado durante el desarrollo del presente Trabajo Fin de Grado.

A mis amigos, por estar siempre presentes.

A mis profesores, por despertar y enriquecer mi interés por las telecomunicaciones.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

UART: Universal Asynchronous Receiver-Transmitter

OPTI: Observatorio de Prospectiva Tecnológica Industrial

OpenGL: Open Graphics Library

IDE: Integrated Development Environment

fps: fotogramas por segundo

GLUT: OpenGL Utility Toolkit

FENIN: Federación Española de Empresas de Tecnología Sanitaria

CPR: Cycles Per Revolution

CMI: Cirugía Mínimamente Invasiva

CDC: Communications Device Class

CASEIB: Congreso Anual de la Sociedad Española de Ingeniería Biomédica

ASCRS: American Society of Colon and Rectal Surgeons

Capítulo 1

Introducción

1.1. Contexto

La laparoscopia es una disciplina quirúrgica de aplicación en la cavidad abdominal que forma parte de las conocidas como técnicas de Cirugía Mínimamente Invasiva (CMI) o de mínimo abordaje. Estas se caracterizan por utilizar varias incisiones (denominadas "puertos") de tamaño reducido para insertar los instrumentos quirúrgicos especializados necesarios para la operación.(MAPFRE Salud, 2016; ASCRS, 2016)

La Cirugía Mínimamente Invasiva se encuadra dentro de la historia moderna de la medicina y ha supuesto una revolución en el ámbito quirúrgico, tanto por la rapidez de su desarrollo como por su aceptación universal. Frente a los procedimientos tradicionales de cirugía abierta, la CMI permite disminuir complicaciones como el dolor posquirúrgico, mejorar la respuesta inmunológica del paciente, reducir la respuesta inflamatoria sistémica asociada con la cirugía y acortar los períodos de recuperación y cicatrización. Además, la implantación de la CMI tiene también un impacto en política sanitaria al disminuir los costes por internación, las infecciones nosocomiales (intrahospitalarias) y las listas de espera. (García Berro, M. y Toribio, C., 2004)

A pesar de las ventajas que presenta la CMI en general, los procedimientos son más complejos que los convencionales y requieren nuevas habilidades quirúrgicas por parte del colectivo médico. En el caso concreto de la cirugía laparoscópica, se ha observado que la adquisición de las destrezas necesarias muestra una curva de aprendizaje superior a la de la cirugía abierta. Las principales razones por las que la laparoscopia presenta mayor complejidad comprenden las dificultades con la percepción espacial y la sensación de profundidad, la carencia de sensación táctil directa (la percepción se hace a través del instrumental quirúrgico) y el efecto de inversión de movimientos o efecto *fulcrum*. (Sánchez-Peralta, L.F. et al., 2010; Sánchez-Margallo, Juan A. et al., 2014; García Berro, M. y Toribio, C., 2004)

1.2. Motivación

Como resultado de lo expuesto en la Sección 1.1, se deduce que el aprendizaje y entrenamiento, tanto de los cirujanos como del personal médico auxiliar, es crucial para asegurar buenos resultados en cirugía laparóscopica. Para ello, los simuladores quirúrgicos suponen una alternativa fiable y segura frente a los riesgos, costes y

cuestionamientos éticos derivados de otros métodos como la tutoría intraoperatoria o las pruebas con animales. (Tarco-Delgado, R. et al., 2007; Sánchez-Peralta, L.F. et al., 2010)

La motivación del presente Trabajo Fin de Grado surge como respuesta a esta necesidad latente de formación y entrenamiento quirúrgico y ante la disponibilidad de una caja entrenadora laparoscópica, cuyas funciones esenciales se vieron mermadas debido a desperfectos y deterioro en parte de sus elementos. Haciendo uso de dicha estructura, se plantea la posibilidad de desarrollar el software y hardware necesario para implementar un simulador laparoscópico completamente útil y funcional.

En este Trabajo Fin de Grado se presenta una primera aproximación al problema planteado, consistente en un sistema de visualización del movimiento de los instrumentos laparoscópicos. Dicho sistema estará constituido por tres componentes fundamentales: la caja entrenadora mencionada, una tarjeta de desarrollo Arduino DUE y un ordenador con sistema operativo de tipo Linux.

1.3. Objetivos

Con el desarrollo y planteamiento iniciales, se busca la consecución de los siguientes objetivos:

- Comprender los elementos principales que conforman el simulador laparoscópico, la manera en la que estos interactúan y su funcionalidad dentro del sistema.
- Analizar las necesidades hardware y software del sistema deseado, intentando adaptar de manera óptima los recursos disponibles a las especificaciones y requisitos temporales del sistema.
- Implementar un subsistema de adquisición, adaptación y medida de las señales de los encoders fiable y funcional.
- Implementar los módulos software de representación bidimensional y tridimensional, buscando el compromiso entre la calidad de los gráficos y la latencia de los datos.

Capítulo 2

Entrenador laparoscópico

2.1. Estudio del simulador laparoscópico

2.1.1. Estructura

El entrenador laparoscópico al que se hace referencia en el presente Trabajo Fin de Grado está formado por una caja metálica y dos instrumentos tubulares acoplados a la misma (ver Figura 2.1).

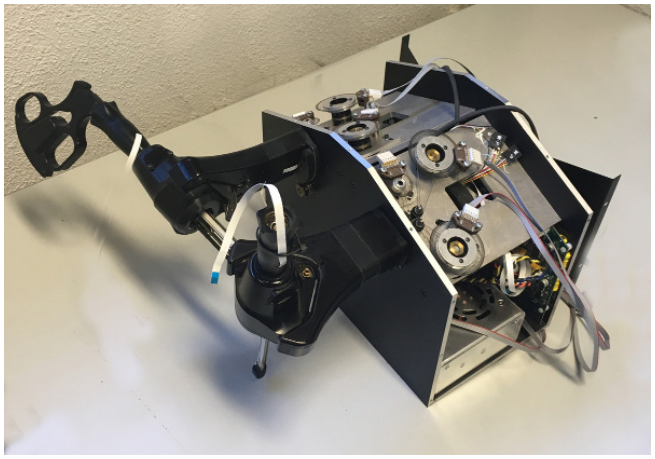

Figura 2.1: Entrenador laparoscópico

La caja metálica constituye el núcleo de adquisición y procesado de la señal. En su interior están alojados ocho encoders rotatorios para el cálculo de la posición, así como motores para las necesidades futuras de realimentación háptica y circuitos eléctronicos específicos de las funcionalidades primitivas del simulador original. Por otro lado, los dos instrumentos tubulares hacen las veces de pinzas laparoscópicas y permiten la interacción directa del usuario con el sistema.

Para el sistema de visualización que nos concierne, no se hará uso de los motores ni de la circuitería electrónica del simulador original. Se utilizarán las señales generadas por los encoders al mover los instrumentos laparoscópicos y el procesado de las mismas tendrá lugar en una tarjeta de desarrollo (ver Sección 2.2.2) externa a la caja laparoscópica. Además, tampoco será posible implementar funcionalidad alguna

relacionada con el sistema de apertura y cierre de las pinzas laparoscópicas, al encontrarse estas en mal estado y requerir un estudio, fuera de los objetivos de este TFG, para su restauración.

Bajo estas consideraciones previas, puede observarse en la Figura 2.2 una ilustración del simulador laparoscópico con los elementos que poseen relevancia e interés en el desarrollo del presente Trabajo Fin de Grado.

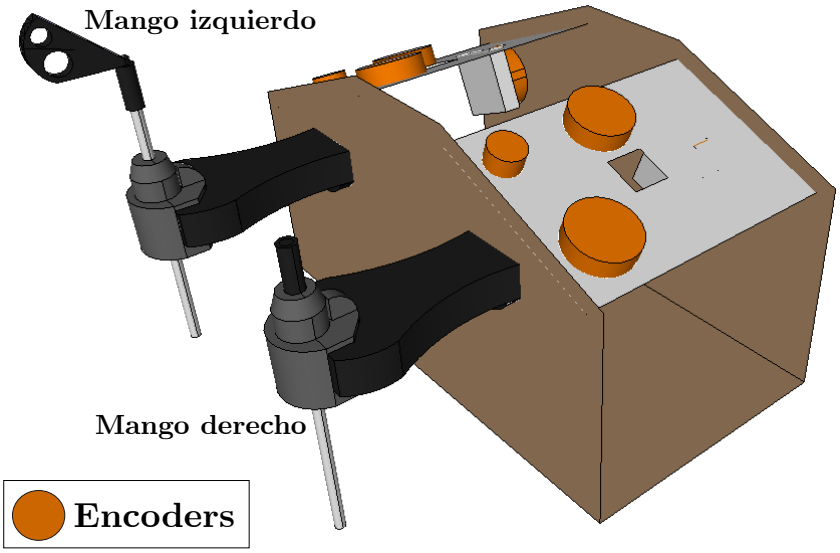

Figura 2.2: Esquema del entrenador laparoscópico utilizado y los elementos relevantes.

2.1.1.1. Grados de libertad

Al disponer de cuatro encoders asociados a cada uno de los instrumentos tubulares, podemos distinguir hasta cuatro movimientos distintos, cada uno de ellos con dos sentidos opuestos. En la Tabla 2.1 están recogidos los ocho movimientos posibles, representados gráficamente en la Figura 2.3.

Número	Movimiento
1	Instrumento izquierdo: dentro/fuera
2	Instrumento izquierdo: arriba/abajo
3	Instrumento izquierdo: izquierda/derecha
4	Instrumento izquierdo: rotación
5	Instrumento derecho: arriba/abajo
6	Instrumento derecho: izquierda/derecha
7	Instrumento derecho: dentro/fuera
8	Instrumento derecho: rotación

Tabla 2.1: Leyenda de los movimientos representados en la Figura 2.3

Las descripciones de la Tabla 2.1 se refieren al movimiento que describen los extremos de los instrumentos tubulares opuestos al mango, que se corresponderían en una operación real con las puntas de las pinzas laparoscópicas. Como se ha mencionado en la Sección 1.1, los movimientos de los dos extremos son inversos, al estar el laparoscopio constituido como una palanca. En el caso de una intervención quirúrgica real, el punto de apoyo de la palanca o fulcro se corresponde con el trócar, instrumento utilizado para crear una abertura en el cuerpo y proporcionar un puerto de acceso durante la cirugía.(Laparoscópica, 2016)

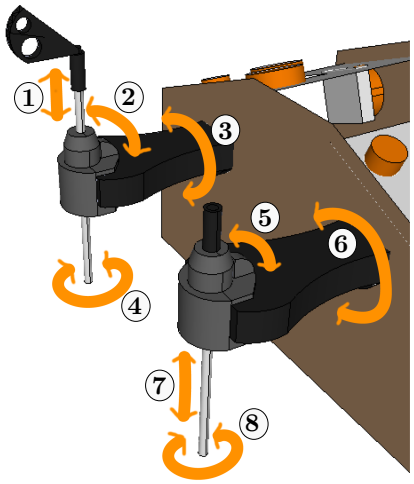

Figura 2.3: Movimientos permitidos por el entrenador laparoscópico.

2.1.2. Encoders

Los encoders son transductores rotativos que transforman un movimiento angular en una serie de impulsos digitales. Para codificar el movimiento de los dos instrumentos tubulares, se utilizan encoders ópticos en cuadratura. Este tipo de encoders dispone de un disco solidario al elemento cuya posición se desea medir, que tiene un patrón radial alternado de líneas opacas y espacios transparentes. Además, se tiene un cabezal de lectura fijo, formado por dos pares led-fotodetector desplazados un cuarto de ranura el uno del otro; estos generan dos pulsos eléctricos A y B desfasados 90º entre sí al detectar las variaciones de luz que resultan del desplazamiento del disco.(Eltra, SILGE Electrónica S.A., 2008; Venegas Requena, 2009)

2.1.2.1. Especificaciones

Los encoders utilizados pertenecen a las series HEDS-9000 y HEDS-9100 de Avago Technologies [1]. Constan de una fuente de luz altamente colimada (un LED) y un circuito integrado detector, consistente en un array de fotodetectores. Estos módulos se usan de manera conjunta con los discos codificados de 2" y 1" de diámetro exterior

[1]Encoders Datasheet: `http://www.avagotech.com/docs/AV02-1867EN`

de US Digital [2], que tienen resoluciones de 2048 CPR y 1024 CPR respectivamente. En la Tabla 2.2 se recogen algunas de las especificaciones más relevantes de los módulos de los encoders mencionados y puede observarse que los discos utilizados coinciden con las recomendaciones dadas por el fabricante.

Modelo encoder	**HEDS-9000**	**HEDS-9100**
Vcc recomendada	de 4.5 V a 5.5 V	de 4.5 V a 5.5 V
Vcc permitida	de -0.5 V a 7 V	de -0.5 V a 7 V
Tensión de salida	de -0.5 V a Vcc	de -0.5 V a Vcc
Option code	U00: Disco recomendado de 2048 CPR	J00: Disco recomendado de 1024 CPR

Tabla 2.2: Especificaciones de los encoders HEDS-9000 y HEDS-9100 utilizados

Hay que tener en cuenta que cada ciclo de cuadratura contiene cuatro transiciones: un flanco de subida y uno de bajada por cada una de las señales A y B. (Gurley Precision Instruments, 2002), tal y como puede observarse en la Figura 2.4. Por tanto, utilizando una decodificación x4, la resolución final de la posición seguirá la Ecuación 2.1.

$$\frac{Posiciones}{Revolución} = 4 \times CPR \tag{2.1}$$

De este modo, con los módulos utilizados se obtendrán las siguientes resoluciones:

$$HEDS9000[Posiciones/Revolución] = 4 \times 2048 = 8192 \tag{2.2}$$

$$HEDS9100[Posiciones/Revolución] = 4 \times 1024 = 4096 \tag{2.3}$$

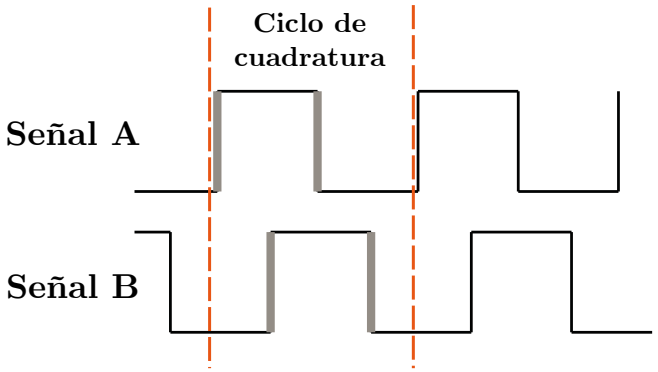

Figura 2.4: Señales A y B de un encoder.

[2] Disk-2 Datasheet: `http://cdn.usdigital.com/assets/datasheets/DISK-2_datasheet.pdf?k=636038441393542057`
Disk-1 Datasheet: `http://cdn.usdigital.com/assets/datasheets/HUBDISK-1_datasheet.pdf?k=636038437936261552`

2.1.2.2. Distribución de los encoders en el simulador laparoscópico

Como se ha mencionado, en la caja entrenadora encontramos dos modelos distintos de encoders. Dos encoders de la serie HEDS-9100 se utilizan para medir la rotación de los instrumentos tubulares y seis encoders del modelo HEDS-9000, para medir el resto de movimientos. En la Tabla 2.3 están recogidos el modelo y movimiento asociado a cada uno de los encoders, representados gráficamente en la Figura 2.5

Número	Movimiento	Modelo
1	Instrumento derecho: rotación	HEDS-9100
2	Instrumento derecho: arriba/abajo	HEDS-9000
3	Instrumento derecho: dentro/fuera	HEDS-9000
4	Instrumento derecho: izquierda/derecha	HEDS-9000
5	Instrumento izquierdo: dentro/fuera	HEDS-9000
6	Instrumento izquierdo: rotación	HEDS-9100
7	Instrumento izquierdo: arriba/abajo	HEDS-9000
8	Instrumento izquierdo: izquierda/derecha	HEDS-9000

Tabla 2.3: Leyenda de los encoders representados en 2.5

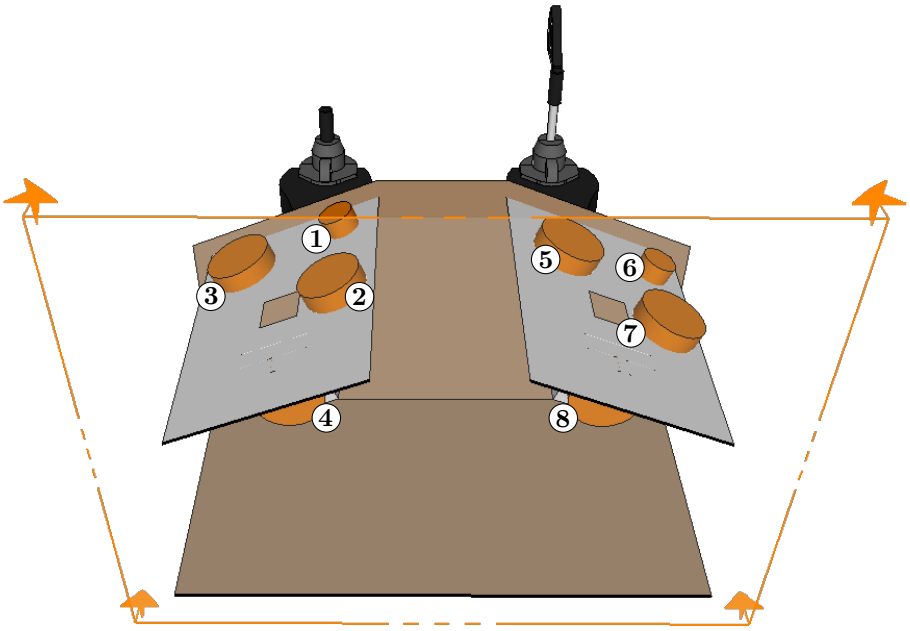

Figura 2.5: Encoders en el entrenador laparoscópico.

2.2. Necesidades hardware y software del sistema

La implementación a desarrollar tiene que dar respuesta a las siguientes necesidades, demandadas para el correcto funcionamiento del sistema de visualización en

tiempo real del movimiento del simulador laparoscópico:

- Detectar los cambios de posición de cada uno de los instrumentos, contemplando los distintos grados de movimiento y estableciendo su valor y sentido.
- Procesar los valores de posición y enviarlos, ajustándose a los tiempos demandados, para su tratamiento por los distintos programas de visualización.
- Decodificar las tramas recibidas para representar en una misma gráfica bidimensional los valores de posición versus tiempo, que se actualice ofreciendo la percepción de un movimiento continuado y sincronizado con la realidad.
- Decodificar las tramas recibidas para ofrecer una representación tridimensional simplificada de los dos instrumentos laparoscópicos, cuya posición se actualice ofreciendo la percepción de un movimiento continuado y sincronizado con la realidad.

Con vistas a cumplir estos requisitos, son necesarios los siguientes elementos hardware y software, cuya aportación a la funcionalidad deseada se analizará en las secciones posteriores:

- Encoders (ver Sección 2.2.1): se utilizarán para determinar la posición de los instrumentos laparoscópicos.
- Arduino Due (ver Sección 2.2.2): se utilizará para recoger las señales generadas por los encoders, procesar los valores de posición y enviarlos al ordenador por el puerto serie.
- Matlab (ver Sección 2.2.3): se utilizará para decodificar las tramas recibidas por el puerto serie y realizar la gráfica bidimensional de posición versus tiempo.
- Chai3d (ver Sección 2.2.4): herramienta de simulación multiplataforma basada en C++, se utilizará para la representación tridimensional del movimiento de los instrumentos tubulares.

La vista general de la arquitectura del sistema completo implementado puede observarse en la Figura 2.6. Las señales de los encoders y la alimentación de los mismos se conectan a la placa Arduino Due con cables jumper. Por otro lado, el Arduino Due se conecta también al ordenador por cable USB a través de su puerto de programación. En el ordenador han de tenerse instalados los recursos necesarios para las gráficas de dos y tres dimensiones: Matlab y Chai3d.

2.2.1. Encoders

Los encoders utilizados han sido los descritos en la Sección 2.1.2, que se encuentran acoplados sobre la estructura del simulador permitiendo registrar los cuatro tipos de movimiento de cada uno de los tubos. Como se puede ver con más detalle en la Sección 3.4.2, ofrecen una resolución adecuada a las necesidades del sistema.

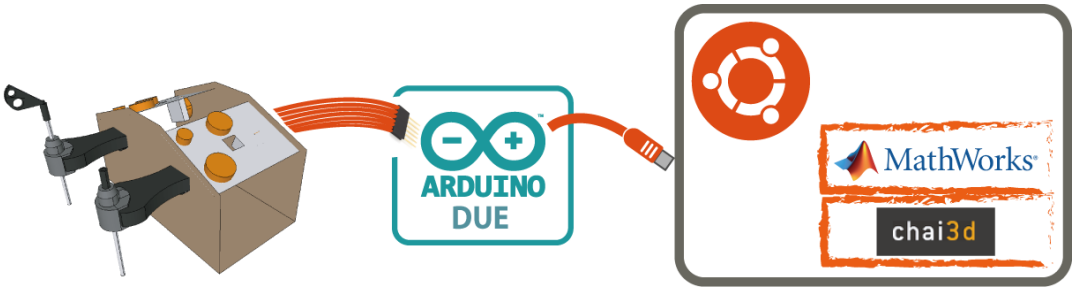

Figura 2.6: Arquitectura general del sistema

2.2.2. Arduino DUE

El Arduino Due es una placa de desarrollo basada en el microcontrolador Atmel SAM3X8E. Utiliza un microcontrolador de 32 bits y frecuencia de reloj de 84 MHz. (Arduino, 2016a)

Dispone de 54 pines digitales I/O y en todos ellos es posible configurar interrupciones (Arduino, 2016c). Por tanto, supone un número de entradas más que suficiente para gestionar las 16 señales provenientes de los encoders: dos señales (canal A y canal B) por cada uno de los ocho encoders.

Por otro lado, el Arduino Due tiene dos puertos USB (Arduino, 2016b) disponibles:

- Puerto nativo: conectado directamente a los pines host USB del microcontrolador SAM3X. Soporta comunicación serie con USB CDC.
- Puerto de programación: conectado a la primera UART del SAM3X a través del ATMEL 16U2, que actúa como conversor USB-Serial.

Resulta importante explicar que la memoria flash del microcontrolador SAM3X tiene que borrarse antes de ser reprogramada; para gestionar dicha tarea, utilizar el puerto de programación supone una opción más fiable. Al comenzar la comunicación por dicho puerto, el ATMEL 16U2 genera automáticamente las señales de borrado y reset antes de cargar el programa desde la UART. Por esta razón, se ha escogido para esta implementación el uso del puerto de programación. Así, el programa de Arduino se reiniciará cuando se abra el puerto serie para su lectura en Matlab o en el programa en C++, sincronizando el comienzo del envío de datos con la recepción de los mismos.

2.2.3. Matlab

El lenguaje y entorno interactivo de Matlab ofrece las herramientas necesarias para la implementación de las gráficas bidimensionales, posibilidad de leer datos del puerto serie, funciones para cambiar el formato de los datos, potencia de cálculo y funciones de trazado.

Se ha tomado la decisión de utilizar Matlab buscando un rendimiento temporal en la actualización de los gráficos mejor al ofrecido por la librería CImg de C++, que fue estudiada como primera opción. Para reajustar la gráfica, desplazándola sobre el eje temporal y mostrando las nuevas medidas tomadas, las funciones ofrecidas por CImg suponían un tiempo de ejecución excesivo al trabajar con la imagen como un conjunto de píxeles, sin distinguir como elementos independientes los ejes, el fondo y

los valores de las muestras. Sin embargo, en Matlab basta con desplazar la referencia de los ejes y añadir las nuevas muestras en el instante temporal correspondiente, sin modificar el trazado de los ejes, el fondo o las muestras previas.

2.2.4. Chai3d

Para la elaboración de la representación tridimensional se ha elegido Chai3d por tratarse de una herramienta con módulos especiales para implementar la realimentación háptica que, si bien no se desplegará en el presente TFG, supone una de las líneas futuras de desarrollo más relevantes y necesarias. Además, se trata de una herramienta que ofrece formas primitivas sencillas (esferas, cilindros, cubos...) y operaciones de transformación geométrica que permiten reducir significativamente los cálculos necesarios para generar los movimientos reales de los instrumentos.

Capítulo 3

Obtención y visualización de datos en tiempo real

3.1. Implementación del subsistema de adquisición y medida de las señales de los encoders

Las claves de la implementación de este subsistema pasan por establecer las conexiones adecuadas entre los encoders y el Arduino Due, gestionar correctamente las interrupciones generadas por las señales A y B en los pines digitales y enviar al ordenador de forma periódica los datos de posición disponibles.

3.1.1. Conexiones encoders - Arduino Due

El patillaje de los encoders de las series HEDS-9000 y HEDS-9100 es el expuesto en la Figura 3.1. Disponen de pines para alimentacion (Vcc), masa (GND) y salidas de los canales A ($Ch.A$) y B ($Ch.B$).

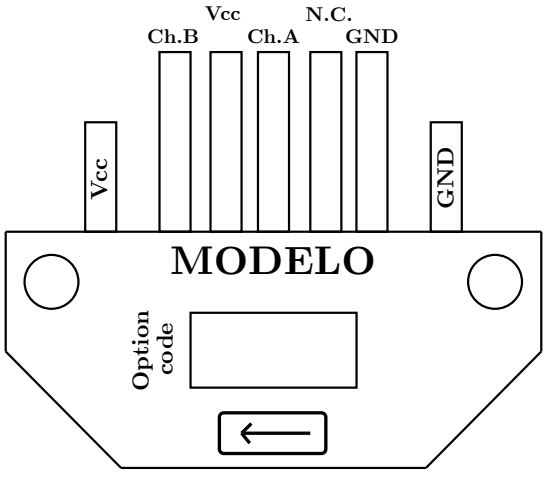

Figura 3.1: Disposición de los pines en los encoders HEDS-9000/9100

Además, se indica también en los encoders el sentido del movimiento cuando la señal A adelanta a la señal B ($\rightarrow$) y la precisión recomendada para el disco codificado (mediante el *option code*).

Una vez conocida la distribución de los pines en los encoders, se describen a continuación las conexiones a realizar entre estos y el Arduino Due:

- **GND:** Se conectarán con los pines GND del Arduino, estableciendo así una misma referencia de tierra.
- **Vcc:** Se conectarán con la salida de 3.3V del Arduino. Si bien 5V serían más recomendables para alimentar a los encoders (ver Tabla 2.2), esto supondría que el nivel alto de las señales A y B sería también de 5V y se aplicarían tensiones mayores de 3.3V en los pines I/O del Arduino, lo cual puede dañar la tarjeta (Arduino, 2016a).
- **Ch.A y Ch. B:** Se conectarán a pines digitales I/O del Arduino.
- **N.C.:** Este pin no tiene función alguna en el encoder utilizado, pero se corresponde con la señal del índice en otros encoders con el mismo encapsulado (como los de las series HEDS-9040/9140). Por tanto, con vistas a futuros cambios y dada la disponibilidad de pines I/O libres, realizaremos las conexiones pertinentes.

3.1.1.1. Adaptación de las señales de los encoders

Para facilitar el tratamiento de las distintas señales y organizar los 40 cables procedentes del entrenador laparoscópico, se ha confeccionado un circuito en una tarjeta de prototipado (ver Figura 3.2). La función de este circuito es reestructurar el entramado de cables: 8 cables tipo bus de 5 hilos recogerán las señales de cada uno de los encoders y las llevarán al circuito de adaptación; una vez allí, se unirán en un solo punto todas las referencias de masa y se hará lo mismo con las entradas dedicadas a la tensión de alimentación.

De este modo, un cable de alimentación (3.3V) y uno de masa (GND) conectados entre la tarjeta y el Arduino bastarán para gestionar las referencias de tensión de todos los encoders, adaptándose a la circunstancia de que el Arduino Due solo dispone de un pin de 3.3V y apenas 4 pines GND.

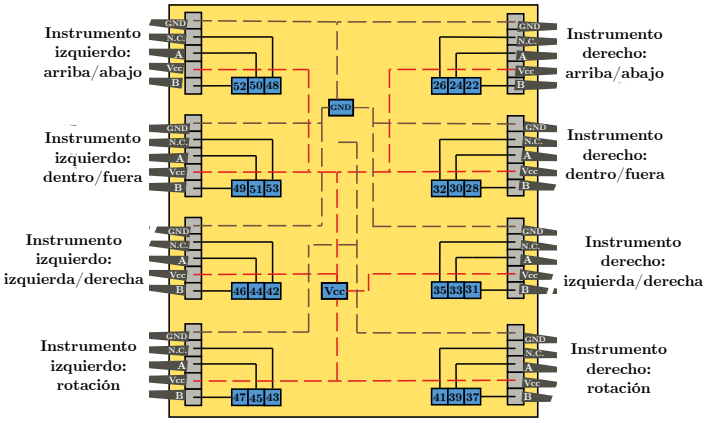

Figura 3.2: Esquema de la placa de acondicionamiento: en los cuadros azules puede observarse el número de pin correspondiente en la placa Arduino Due

3.1.2. Implementación en Arduino

3.1.2.1. Posición y convenio de signos

Los valores de posición con los que trabajará Arduino estarán medidos en *'número de pulsos del encoder'* y serán enviados con tales unidades al ordenador, que será el encargado de realizar las transformaciones pertinentes para obtener los valores de posición lineales (mm) o angulares (^{o}) correspondientes. En cualquier caso, estos siempre serán valores relativos respecto a la posición inicial del instrumento en el momento de comenzar a medir, que tomará valor cero.

El convenio de signos adoptado queda reflejado en la Figura 3.3. Se tomará sentido positivo del movimiento cuando la señal A adelante a la señal B y sentido negativo en caso contrario.

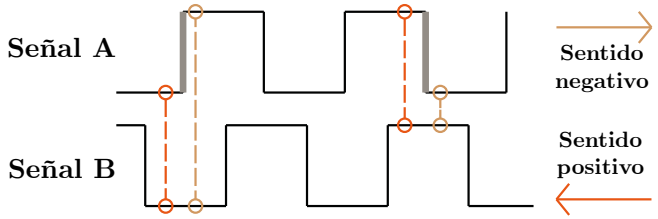

Figura 3.3: Convenio de signos adoptado para los valores de posición.

3.1.2.2. Interrupciones externas

En los pines digitales del Arduino conectados a los canales A o B de alguno de los encoders, se configurarán interrupciones externas que se activarán cuando cambie la señal correspondiente ($CHANGE$), es decir, tanto en los flancos de subida como en los de bajada.

Las rutinas de atención a la interrupción actualizarán la posición de manera coherente con el convenio de signos adoptado:

- **Rutina de atención a la interrupción A:** Se incrementará en una unidad la posición si la lectura de las señales A y B devuelve el mismo valor ($HIGH$ o LOW). En caso contrario, se disminuirá en una unidad.

- **Rutina de atención a la interrupción B:** Se disminuirá en una unidad la posición si la lectura de las señales A y B devuelve el mismo valor ($HIGH$ o LOW). En caso contrario, se incrementará en una unidad.

3.1.2.3. Formato de trama y envío

Como ya se ha comentado en la Sección 2.2.2, se elige como medio de comunicación entre el ordenador y el Arduino Due el puerto serie, conectado al puerto de programación del Arduino. Las tramas se transmitirán con la instrucción `Serial.println()`, que envía los datos por el puerto serie con formato de texto ASCII, seguido de un carácter de retorno y nueva línea ("\r\n"). (Arduino, 2016d)

La transmisión por puerto serie presenta ciertas limitaciones al establecer una comunicación asíncrona y cuyo límite estándar de velocidad en UNIX está establecido en 115200 bps (Negus, C. y Weeks, T., 2001; Linurs, 2016; Die.net, 2016; The Linux Documentation Project, 2011). No obstante, se trata de la forma menos compleja de transmitir datos entre el Arduino y el ordenador y resulta útil para realizar una primera aproximación al problema planteado.

Se utilizará la configuración **115200 8N1** para el puerto serie en modo asíncrono, esto es: tasa de 115200 bps (*115200*), 1 bit de inicio (presente en todas las configuraciones), 8 bits de datos (*8*), ningún bit de paridad (*N*) y 1 bit de parada (*1*). El formato de la trama puede observarse en la Tabla 3.1

1	8	1
Start bit	Data	Stop bit

Tabla 3.1: Formato de las tramas de envío

En cuanto al formato de los datos (ver ejemplo de trama en la Tabla 3.2), los valores de posición se enviarán como la codificación ASCII extendida de cada una de sus cifras, precedidos de un identificador. La instrucción `Serial.println()` tomará estas tramas de datos, las fragmentará en bytes y añadirá los bits de inicio y de parada correspondientes antes de enviarlas al puerto serie.

Trama	a	7	2	1	5	... →
ASCII	97	55	50	49	53	... →
Bits	01100001	00110111	00110010	00110001	00110101	... →
→ ...	h	-	8	3	\r	\n
→ ...	104	45	56	51	13	10
→ ...	01101000	00101101	00111000	00110011	00001101	00001010

Tabla 3.2: Ejemplo de trama de datos, conversión a ASCII y cadena de bits correspondiente.

Los identificadores son carácteres ASCII (extendido) de la *a* a la *h* que se relacionan de manera unívoca con los distintos tipos de movimiento posibles, según se referencia en la Tabla 3.3.

El uso de la codificación ASCII extendida se apoya en las siguientes consideraciones:

- Coherencia con el funcionamiento de `Serial.println()`, que envía los datos en formato ASCII.
- Facilidad para distinguir los identificadores (caracteres alfabéticos) de los valores de posición (caracteres numéricos)
- Si bien el formato ASCII de 7 bits es suficiente para el rango de valores que

Identificador	Instrumento	Movimiento
a	Derecho	Rotación
b	Izquierdo	Rotación
c	Derecho	Dentro/fuera
d	Izquierdo	Dentro/fuera
e	Derecho	Arriba/abajo
f	Izquierdo	Arriba/abajo
g	Derecho	Izquierda/derecha
h	Izquierdo	Izquierda/derecha

Tabla 3.3: Identificadores y movimientos asociados

necesitamos enviar (caracteres imprimibles y de control), el formato de 8 bits ofrece mejor compatibilidad con las configuraciones estándar de envío (8N1) y las funciones de lectura de distintos lenguajes (en este caso Matlab o C), que permiten la lectura byte a byte.

- Si transmitiéramos los datos en formato binario, tendríamos que fijar de una forma más restrictiva el tamaño de la trama. El uso de codificación ASCII permite utilizar una longitud variable para los valores de posición en función del número de cifras, adaptando el tamaño de la trama al rango de la medida tomada en lugar de ajustarlo en función del rango posible de las medidas.

3.2. Sistema en tiempo real

El envío de las señales medidas no se efectuará de forma continua en el tiempo, sino que ha de elegirse un período de muestreo T, buscando un compromiso entre el tiempo empleado en el envío y procesado de las señales y el tiempo necesario para una realimentación gráfica y háptica efectiva.

3.2.1. Requisitos de tiempo real

Considerando futuras necesidades de realimentación háptica, habría que tener en cuenta que esta presenta una capacidad de detección típica de 1 kHz, con un pico de sensibilidad en torno a los 300 Hz. Por tanto, una frecuencia de muestreo igual o superior a 1 kHz permitiría lograr una percepción realista, al mismo tiempo que aseguraría la estabilidad del lazo de realimentación háptica. (Kuchenbecker et al., 2006; Ni et al., 2014)

Sin embargo, la parte relativa a la visualización del movimiento presenta unas exigencias de tiempo menos restrictivas; por tanto, el procesado de la imagen y la realimentación háptica podrían realizarse de manera simultánea, pero con períodos distintos adaptados a las demandas de cada proceso.(Gagalowicz, A. y Philips, W., 2005)

Para establecer qué frecuencias pueden considerarse adecuadas para el sistema de visualización, hay que tener en cuenta dos fenómenos fundamentales en la percepción humana del movimiento:

- **Fenómeno phi**: consiste en la ilusión óptica de percibir como movimiento continuo una secuencia de imágenes estáticas, siempre que la frecuencia de visionado de las mismas sea suficientemente alta y las distancias recorridas por los objetos en movimiento suficientemente pequeñas. (Lewis, 2012) Algunos de los rangos de frecuencia comúnmente aceptados como índice de la calidad de la percepción se listan a continuación (Bakaus, 2014):

  - Por debajo de 10-12 Hz: los fotogramas se reconocen como imágenes estáticas individuales, sin percepción de movimiento continuo.
  - Alrededor de 24 Hz: valor mínimo tolerable para una adecuada percepción de movimiento continuo.
  - En torno a 60 Hz: punto óptimo para la mayoría de las personas, si bien valores mayores pueden suponer mejoras en el contraste y nitidez de la imagen.

  En lo que a este trabajo respecta, estas frecuencias serán las responsables de las restricciones en el período de muestreo de las señales de los encoders.

- **Frecuencia crítica de fusión**: se trata de la frecuencia mínima a partir de la cual una estimulación luminosa intermitente pasa de percibirse como una fluctuación luminosa (*flicker* o parpadeo) a producir una impresión estable continua (Piéron, 1990).

  Este umbral toma distintos valores en función de las características de la propia imagen (fotogramas con más brillo presentan un valor más alto) y de las condiciones de visionado (la visión periférica tiene mayor sensibilidad temporal que la central y, por tanto, un umbral de flicker más alto), si bien suele situarse entre 50 y 90 Hz (Poynton, 2012; Davis et al., 2015).

  Estos rangos de percepción impondrán restricciones en la frecuencia de actualización de las imágenes.

3.2.2. Implementación del subsistema de adquisición de medidas conforme a los requisitos de tiempo

Con vistas a cumplir las condiciones temporales exigidas, se han seguido los siguientes pasos:

- Medida de los tiempos máximos de ejecución de las distintas tareas implicadas en el funcionamiento global del sistema (ver Sección 3.2.2.1).

- Establecimiento de los períodos de muestreo y actualización de las imágenes, en función de los requerimientos y los tiempos de ejecución medidos.

- Configuración de un timer en Arduino Due para generar una interrupción con la frecuencia de muestreo. En la rutina de atención a la interrupción se activará un flag para indicar que las tramas de datos han de enviarse al ordenador.

3.2.2.1. Medidas de tiempos

Para poder planificar los períodos de muestreo de los datos, se han tomado medidas de los procesos que siguen:

- Encapsulado de la trama y envío por puerto serie en Arduino: se ha utilizado la función `micros()`. Se ha prescindido del timer fijado para el período de muestreo durante las series de medidas, al interferir este con el correcto funcionamiento de `micros()`.
- Procesado de una trama en Matlab: se han utilizado las funciones `tic` y `toc`. Incluye el tiempo de lectura y decodificación de la trama, así como el tiempo empleado en actualizar la gráfica con el valor de la muestra recibida.
- Procesado de una trama en Chai3d: se ha utilizado la función `clock_gettime` con el reloj *CLOCK_MONOTONIC*. Incluye el tiempo de lectura y decodificación de la trama.
- Cálculo de las posiciones de los objetos tridimensionales y actualización de la gráfica en chai3d: se ha utilizado la función `clock_gettime` con el reloj *CLOCK_MONOTONIC*.

En la Tabla 3.4 pueden observarse los tiempos de ejecución obtenidos a partir de las mediciones efectuadas. En el caso de las medidas tomadas en Matlab, se ofrecen los porcentajes obtenidos a partir de 200 medidas. En el resto de casos, se trata de los valores máximos obtenidos tras 20 segundos de ejecución.

Entorno	**Tarea**	**Tiempo de ejecución**	
Arduino	Comunicación con Matlab	$t_{máx}$:	4372 μs
Arduino	Comunicación con Chai3d	$t_{máx}$:	2477 μs
Chai3d	Posiciones y gráfica	$t_{máx}$:	12132 μs
Chai3d	Procesado de trama	$t_{máx}$:	41129 μs
Matlab	Procesado de trama	50 %:	$<$ 10000 μs
		99 %:	$<$ 30000 μs
		100 %:	$<$ 40000 μs

Tabla 3.4: Tiempos de ejecución de los distintos procesos

En base a los tiempos obtenidos, resulta inmediato pensar que es necesario otro medio de transmisión más rápido u otra forma de codificación más eficiente para cumplir los requisitos de frecuencia del lazo de realimentación háptica. Sin embargo, las medidas resultan adecuadas para implementar los sistemas de visualización.

Con el fin de aumentar la eficiencia, ya se decidió enviar en una sola trama todas las medidas frente al planteamiento inicial en el que cada medida se enviaba en una trama; así, se consigue una mejor eficiencia enviando 1 solo carácter de fin de trama de datos ("\n") en lugar de 8. Otras medidas tomadas en función de la finalidad última del programa se comentarán en las secciones posteriores.

3.3. Módulo software de gráficas bidimensionales

En la presente sección se referenciarán las decisiones de diseño tomadas para la representación del valor de la posición de los ocho encoders y su variación en tiempo real.

3.3.1. Temporización

En base a las mediciones efectuadas en la Sección 3.2.2.1, se ha tomado la decisión de establecer un período de muestreo y envío de las señales de posición $T = 30ms$, lo que supone una frecuencia $f = 33,33Hz \simeq 33Hz$.

Desde Arduino Due se enviarán los datos con la frecuencia mencionada y Matlab trabajará en un bucle continuo, procesando los datos tan pronto como los reciba por el puerto serie. Dado que casi el 99 % de las veces el tiempo de procesamiento de una muestra en Matlab es menor que el período establecido, no se producirán retardos significativos y estos no se acumularán, lo cual también se ha comprobado empíricamente.

El valor de frecuencia elegido resulta admisible para una percepción de movimiento continuo, si bien no es suficiente para evitar el *flickering* o parpadeo (según lo estipulado en la Sección 3.2.1). Este parpadeo se soluciona parcialmente con la funcionalidad `DoubleBuffer`, activada por defecto para todas las figuras en Matlab. El *double buffer* consiste en lo siguiente: se tiene una pantalla oculta (*back buffer*), que es una copia de la pantalla principal mostrada al usuario (*primary surface*); a la hora de refrescar la gráfica, se actualiza la imagen de la pantalla oculta y no se muestra dicho contenido en la pantalla principal hasta que no termina el procesado y pintado de la imagen completa. (Altman, 2015)

Experimentalmente se verifica la ausencia de *flickering* en las gráficas de las señales, aunque sí se observa cierto parpadeo en el eje de abcisas, al no ser la propiedad `DoubleBuffer` aplicable a los ejes.

3.3.1.1. Variabilidad de la longitud de las tramas

Para reducir el tiempo empleado en la transmisión de datos, se ha optado por una reducción de la longitud de la trama en función de los cambios en la posición de los instrumentos. Así, si el valor de una determinada posición no se ha modificado respecto al valor enviado por última vez, se omite en la trama la parte correspondiente a dicha posición.

3.3.2. Implementación en Matlab

El programa realizado presenta ocho gráficas superpuestas, identificadas en la leyenda de la esquina superior derecha, tal y como se muestra en la Figura 3.4. El eje de ordenadas representa el número de pulsos del encoder y el eje de abcisas, el instante de muestreo.

El eje de abcisas se desplaza al ritmo marcado por la recepción de nuevas muestras. Se presentan en la ventana gráfica 100 instantes de muestreo distintos; solo los 50 primeros presentan el valor de posición correspondiente, mientras que los 50 siguientes están en blanco y representan valores futuros por muestrear.

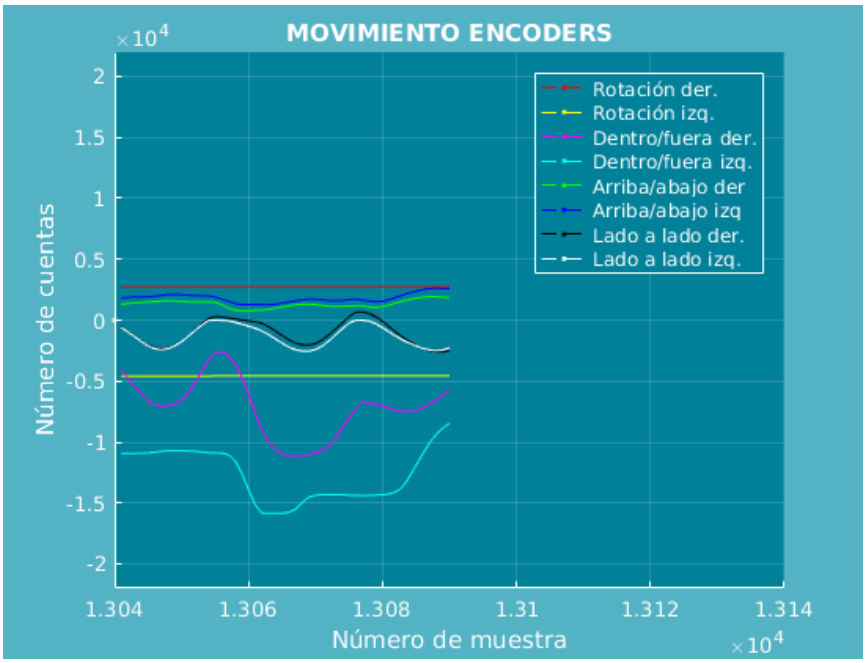

Figura 3.4: Captura de pantalla de las gráficas bidimensionales en tiempo real implementadas en Matlab

En cuanto al eje de ordenadas, permanece fijo durante toda la ejecución del programa, con valores de escala mínimo y máximo situados en ±22000. Estos valores se han establecido en base a la Tabla 3.5; en ella se muestran, para cada tipo de movimiento, las siguientes medidas:

- **Máximo valor de posición:** medido como el máximo número de pulsos obtenido al mover el instrumento dentro del rango que su estructura mecánica permite. Expresado en valor absoluto.
- **Error máximo en el valor de posición:** medido como el máximo número de pulsos obtenido tras mover el instrumento partiendo de una posición inicial conocida y regresando a la misma. Expresado en valor absoluto.

A la hora de desarrollar el programa, ha tenido que prestarse especial atención a dos aspectos que lastraban el cumplimiento de los requisitos de tiempo:

- **Uso de leyendas:** siguiendo el uso habitual, cada vez que se añade una nueva muestra a la gráfica y se modifican los ejes, se actualiza también la leyenda asociada a estos, con la consiguiente carga en el tiempo de ejecución que ello supone. Por esta razón, se ha adoptado la medida de crear una leyenda estática, asociada a unos ejes ocultos creados al inicio del programa y que no son modificados en ningún momento. (Altman, 2015)
- **Uso de `plot`:** el uso de la función `plot` en un bucle continuo cada vez que debe pintarse una nueva serie de muestras funciona correctamente cuando el número de iteraciones es pequeño. No obstante, para un funcionamiento prolongado en el tiempo, la acumulación de muestras implica un aumento en el consumo de memoria que ralentiza el programa. Para este tipo de gráficas en las que

Identificador de movimiento	**Valor máximo de posición** [nº de pulsos]	**Error absoluto** [nº de pulsos]
a	6171	13
b	6175	2
c	21542	44
d	21529	3
e	9474	5
f	9442	9
g	9754	8
h	9668	11

Tabla 3.5: Valor máximo de posición y error absoluto máximo para cada uno de los movimientos, definidos por los identificadores expuestos en la Tabla 3.3

se pretende crear animaciones de líneas a partir de la acumulación de datos recibidos de forma continua, Matlab ofrece la función `animatedLine` con un mejor rendimiento.

Por último, cabe mencionar la particularidad de que Matlab solo reconoce automáticamente nombres de puerto serie de la forma "dev/ttyS$[0-255]$". Es necesario, por tanto, crear un enlace simbólico con el formato de nombre admitido que apunte al puerto serie al que Arduino está transmitiendo los datos.(MathWorks Support Team, 2016)

3.4. Módulo software de representación tridimensional del instrumento

En la presente sección se referenciarán las decisiones de diseño tomadas para la representación tridimensional de los dos instrumentos tubulares y su variación en tiempo real.

3.4.1. Temporización

En base a las mediciones efectuadas en la Sección 3.2.2.1, se ha tomado la decisión de establecer un período de muestreo y envío de las señales de posición $T = 42ms$, lo que supone una frecuencia $f = 23,81Hz \simeq 24Hz$.

Desde Arduino Due se enviarán los datos con la frecuencia mencionada y Chai3d trabajará en un bucle continuo con varias hebras funcionando en paralelo. Las tareas a ejecutar de forma simultánea son el procesado de los datos recibidos por el puerto serie, el cálculo de la posición de los objetos tridimensionales en base a dichos datos y la actualización de la pantalla. Dado que el tiempo que se tarda en procesar una muestra es menor que el período establecido, no se acumularán retardos que ralenticen el programa.

El valor de frecuencia elegido resulta admisible para una percepción de movimiento continuo, si bien no es suficiente para evitar el *flickering* o parpadeo. Del mismo modo que en la Sección 3.3.1, el parpadeo se soluciona implementando el *double*

buffer mediante una correcta configuración inicial de la ventana (usando el modo `GLUT_DOUBLE`) y el uso de la función `glutSwapBuffers`() como último paso en la actualización de la gráfica. Experimentalmente se verifica la ausencia de *flickering*.

3.4.2. Relación entre el número de pulsos y el desplazamiento angular y lineal correspondiente

Para cada uno de los tipos de movimiento, se han realizado 10 medidas del número de pulsos correspondiente a un desplazamiento angular (°) o lineal (cm) conocido. Se consideran las siguientes asunciones:

- Los dos instrumentos tubulares presentan idénticas características, tanto en lo relativo a sus dimensiones como en los movimientos que describen. Por lo tanto, las ecuaciones que rijan dichos movimientos serán iguales en ambos casos.

- El desplazamiento o rotación del instrumento y el número de pulsos son directamente proporcionales. La constante de proporcionalidad se establecerá en base a las series de medidas efectuadas.

3.4.2.1. Rotación

Se han tomado 5 medidas para una rotación de **90º** y 5 medidas para una rotación de **180º** , tal y como se refleja en la Tabla 3.6.

Grados sexagesimales	**Nº de pulsos**	**pulsos/º**	**º/pulso**
90	3386	37,622	0,027
90	3415	37,944	0,026
90	3348	37,200	0,027
90	3183	35,367	0,028
90	3239	35,989	0,028
180	6153	34,183	0,029
180	6150	34,167	0,029
180	6179	34,328	0,029
180	6185	34,361	0,029
180	6177	34,317	0,029
	Media:	35,548	0,028

Tabla 3.6: Relación entre el ángulo de rotación y el número de pulsos en el movimiento en el movimiento "rotación".

3.4.2.2. Dentro/fuera

Se han tomado 10 medidas para un desplazamiento de **7cm** , reflejadas en la Tabla 3.7.

Centímetros	**Nº de pulsos**	**pulsos / cm**	**cm / pulso**
7	8585	1226,4286	0,0008
7	8632	1233,1429	0,0008
7	8636	1233,7143	0,0008
7	8737	1248,1429	0,0008
7	8500	1214,2857	0,0008
7	8593	1227,5714	0,0008
7	8623	1231,8571	0,0008
7	8641	1234,4286	0,0008
7	8515	1216,4286	0,0008
7	8712	1244,5714	0,0008
Media:	8617,4	1231,0571	0,0008

Tabla 3.7: Relación entre el desplazamiento lineal y el número de pulsos en el movimiento "dentro/fuera".

3.4.2.3. Arriba/abajo

Se han tomado 10 medidas para una rotación de $\mathbf{53,13^o}$, recogidas en la Tabla 3.8.

Grados sexagesimales	**Nº de pulsos**	**pulsos/º**	**º/pulso**
53,13	4938	92,942	0,011
53,13	4956	93,281	0,011
53,13	4941	92,998	0,011
53,13	4967	93,488	0,011
53,13	4953	93,224	0,011
53,13	4937	92,923	0,011
53,13	4943	93,036	0,011
53,13	4951	93,187	0,011
53,13	4949	93,149	0,011
53,13	4942	93,017	0,011
Media:	4947,7	93,124	0,011

Tabla 3.8: Relación entre el ángulo de rotación y el número de pulsos en el movimiento "arriba/abajo".

3.4.2.4. Izquierda/derecha

Se han tomado 10 medidas para una rotación de $\mathbf{23,05^o}$, recogidas en la Tabla 3.9.

Grados sexagesimales	Nº de pulsos	pulsos/º	º/pulso
53,13	2403	104,2516	0,0096
53,13	2407	104,4252	0,0096
53,13	2400	104,1215	0,0096
53,13	2404	104,2950	0,0096
53,13	2401	104,1649	0,0096
53,13	2403	104,2516	0,0096
53,13	2406	104,3818	0,0096
53,13	2403	104,2516	0,0096
53,13	2405	104,3384	0,0096
53,13	2402	104,2082	0,0096
Media:	2403,4	104,269	0,0096

Tabla 3.9: Relación entre el ángulo de rotación y el número de pulsos en el movimiento "izquierda/derecha".

3.4.2.5. Análisis de los resultados

De los resultados obtenidos en las mediciones se desprenden las siguientes conclusiones:

- En el caso de la traslación, la resolución del encoder resulta más que suficiente para una resolución milimétrica del sistema, al obtenerse valores superiores en varios órdenes de magnitud a 1 pulso/mm.

- En el caso de las rotaciones, la resolución del encoder resulta suficiente para una resolución del sistema de décimas de grado sexagesimal, no obteniéndose en ningún caso valores inferiores a 10 pulsos/º.

- La dispersión de los resultados indica la falta de precisión en las medidas realizadas. Para una validación más precisa del instrumento, fuera de los objetivos de este TFG, sería necesario:

    1. Calibrar el instrumento definiendo de forma clara en el entorno real una posición inicial de reposo, mediante marcas o topes mecánicos.
    2. Disponer de los instrumentos adecuados para la medida de los desplazamientos lineales y angulares, ya que los valores de referencia tomados como conocidos resultan poco fiables.
    3. Corregir deficiencias mecánicas en la estructura, tales como la falta de estabilidad de la base o el incorrecto arrollamiento de los hilos metálicos que transmiten el movimiento a los encoders.

- Las constantes de proporcionalidad que serán utilizadas para implementar el programa adecuadamente están resumidas en la Tabla 3.10.

Identificador	Constante de proporcionalidad	
a	35,5	[pulsos / º]
b	35,5	[pulsos / º]
c	1231,1	[pulsos / cm]
d	1231,1	[pulsos / cm]
e	93,1	[pulsos / º]
f	93,1	[pulsos / º]
g	104,3	[pulsos / º]
h	104,3	[pulsos / º]

Tabla 3.10: Constantes de proporcionalidad [pulsos/º] establecidas para cada uno de los movimientos, definidos por los identificadores expuestos en la Tabla 3.3

3.4.3. Variaciones respecto al formato de trama original

Teniendo en cuenta las medidas expuestas en la Sección 3.4.2, se han establecido unas cotas mínimas de variación para cada movimiento, de forma que no se envían los datos relativos a la posición hasta que la diferencia entre el último valor enviado y el valor actual difieran en una cantidad mayor a las cotas fijadas en la Tabla 3.11. En ella, se muestran las cotas en "número de pulsos" y su equivalente en **mm** o º aplicando la constante de proporcionalidad correspondiente definida en la Tabla 3.10.

Además, en las tramas no se enviarán los valores de posición, sino la diferencia entre el último enviado y el del correspondiente instante de muestreo. Así, los movimientos de los objetos en Chai3d se aplicarán sobre su posición actual y no tendrán que almacenarse referencias a la posición inicial.

Identificador de movimiento	Cota mínima [nº de pulsos]	Cota mínima [Aplicando cte. de proporcionalidad]
a	13	0,366 º
b	13	0,366 º
c	60	0,487 mm
d	60	0,487 mm
e	34	0,365 º
f	34	0,365 º
g	38	0,364 º
h	38	0,364 º

Tabla 3.11: Cotas mínimas de variación establecidas para cada uno de los movimientos, definidos por los identificadores expuestos en la Tabla 3.3

3.4.4. Representación tridimensional de los instrumentos

Se ha optado por aproximar cada uno de los instrumentos tubulares laparoscópicos, cuyas dimensiones fundamentales están definidas en la Figura 3.5, por un ortoedro

de $\mathbf{0,6cm \times 0,6cm \times 28cm}$ y una esfera de radio $\mathbf{R = 2cm}$ a modo de punto de apoyo, tal y como se muestra en la Figura 3.6. Las esferas estarán centradas en los puntos $\mathbf{(0, \pm 6'75cm, 0)}$ y los ortoedros, en los puntos $\mathbf{(0, \pm 6'75cm, -2cm)}$.

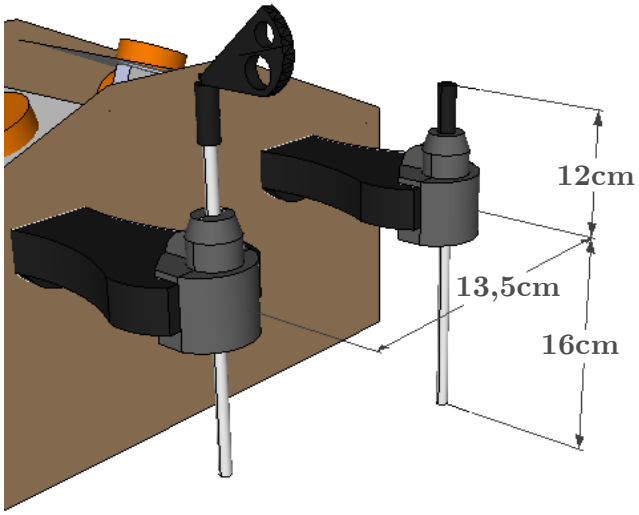

Figura 3.5: Altura y separación entre los instrumentos tubulares

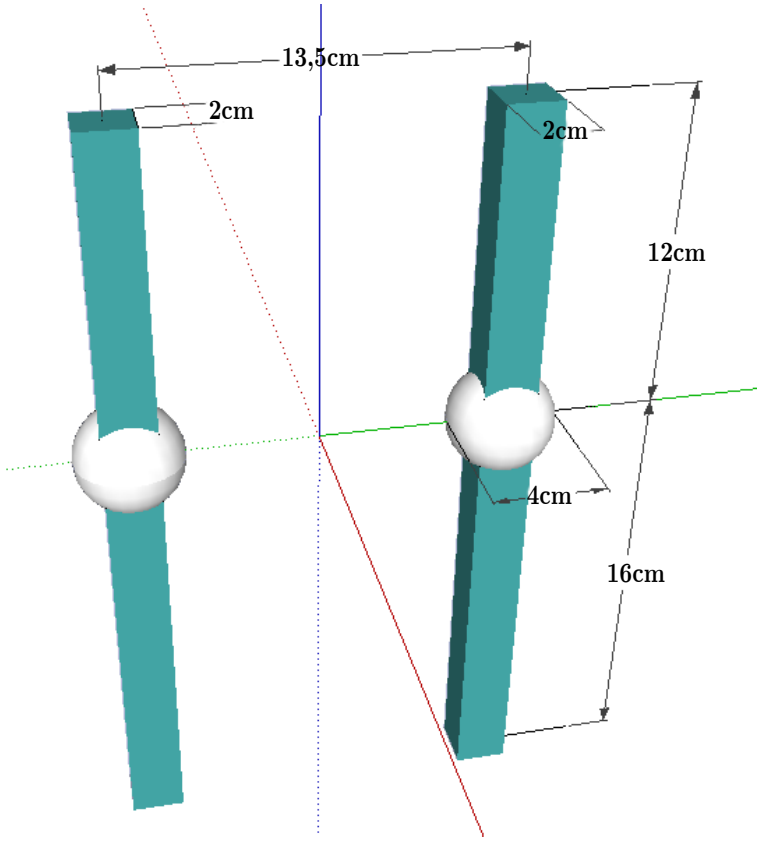

Figura 3.6: Aproximación tridimensional de los instrumentos tubulares

3.4.5. Descripción geométrica de los movimientos

En esta Sección se explicarán sucintamente algunos de los desarrollos geométricos seguidos para conseguir realizar el movimiento deseado de los objetos a partir de las funciones soportadas en Chai3d.

Antes de entrar en detalle con cada uno de los movimientos, merece la pena explicar ciertos detalles que pueden causar confusión:

- **Sistema de coordenadas.** Un mismo objeto puede estar referenciado a distintos sistemas de coordenadas: el *sistema de coordenadas global* es el sistema de coordenadas del espacio 3D completo, común para todos los objetos; el *sistema de coordenadas local* es el sistema de coordenadas asociado a cada uno de los objetos (Martínez Valdéz, A. et al., 2016).

- **Matriz de transformación.** Se define la *matriz de transformación homogénea* como aquella que transforma un vector de posición expresado en coordenadas homogéneas respecto a un sistema de coordenadas que ha sido rotado y trasladado (Castañeda Marín, R., 2016). La *matriz de transformación local* transforma un vector expresado en coordenadas globales al sistema de coordenadas locales.

- **Rotaciones en torno a un eje global.** En este tipo de rotaciones, el eje que describe el sentido y dirección de la rotación está referido al sistema global de coordenadas. No obstante, el centro de rotación se sitúa en el centro del objeto sobre el que se aplica la rotación.

3.4.5.1. Rotación

Para llevar a cabo la rotación del instrumento tubular en torno a su propio eje longitudinal, tal y como se muestra en las Figuras 3.7 y 3.8:

1. El vector normalizado con misma dirección y sentido que el eje longitudinal del ortoedro es el vector (0,0,1) (eje z) referido al sistema de coordenadas local del ortoedro.

2. La matriz de transformación local permite pasar del sistema de coordenadas global al local. Multiplicando el vector (0,0,1) (coordenadas globales) por dicha matriz, obtendremos las coordenadas globales del vector de rotación normalizado.

3. Como el módulo del vector no resulta determinante para la rotación y el eje longitudinal del ortoedro pasa por el centro de la esfera, el mismo vector de rotación referido al sistema de coordenadas local de la esfera será válido.

4. Como la rotación se efectúa sobre un eje global, tenemos que trasladar el vector del sistema de referencia local de la esfera al sistema de referencia global, restando las coordenadas de la esfera.

5. Rotamos el objeto el ángulo deseado en torno al eje calculado. Ha de tenerse en consideración el sentido de giro:

- Para una rotación en sentido horario, visto desde la planta del instrumento, el encoder proporciona un número de pulsos positivo.
- Para un eje con sentido negativo en z, la función `rotateAboutGlobalAxisDeg` de Chai3d rota el instrumento en sentido horario, visto desde la planta del instrumento.
- Como conclusión, el ángulo pasado como parámetro a la función ha de tener signo opuesto al obtenido a partir del número de pulsos del encoder.

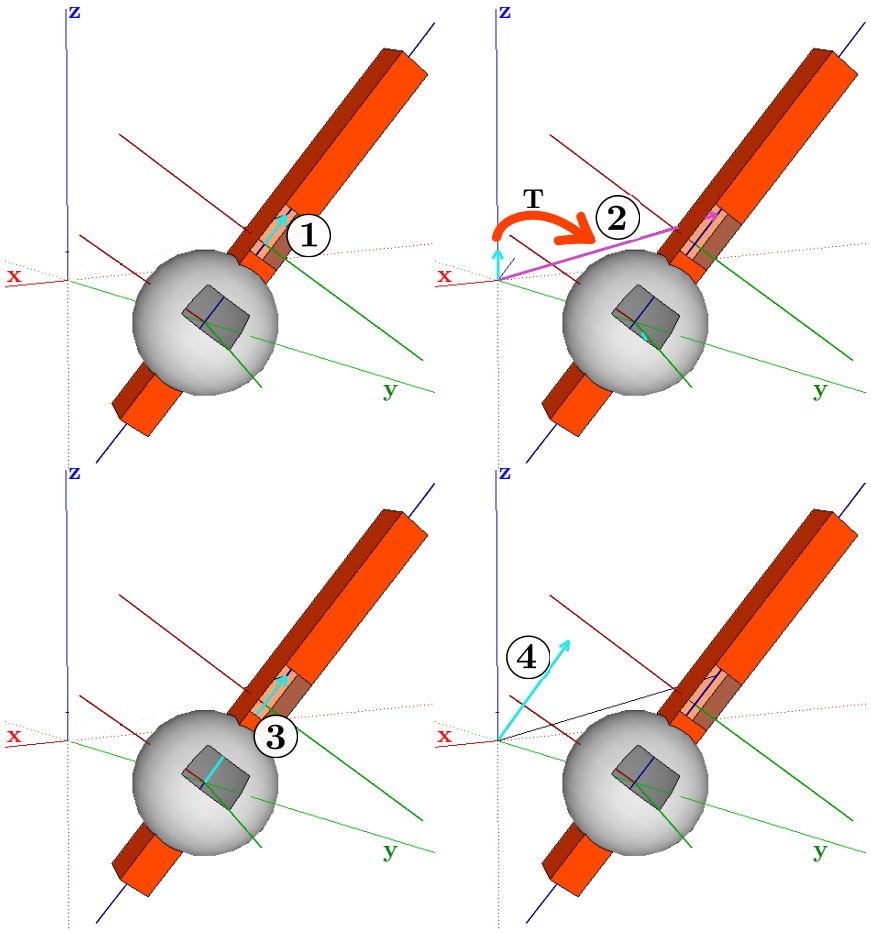

Figura 3.7: Pasos a seguir para la rotación en torno al eje longitudinal

3.4.5.2. Dentro/fuera

Para trasladar el instrumento tubular $\mathbf{z_i}$ **mm** en la dirección de su eje longitudinal, tal y como se muestra en la Figura 3.9:

1. Aplicar el factor de escalado de la representación tridimensional al desplazamiento z_i.
2. El vector de desplazamiento es el vector $(0,0,z_i)$ referido al sistema de coordenadas local del ortoedro.

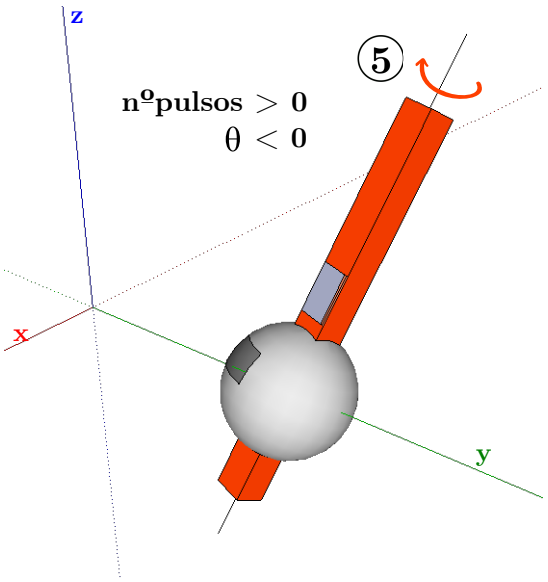

Figura 3.8: Resultado de la rotación en torno al eje longitudinal

3. La matriz de transformación local permite pasar del sistema de coordenadas global al local. Multiplicando el vector $(0,0,z_i)$ (coordenadas globales) por dicha matriz, obtendremos las coordenadas globales del vector de desplazamiento.

4. Para calcular las componentes del offset de la traslación, restamos componente a componente el vector de desplazamiento y el vector de posición del objeto.

5. Trasladamos el objeto el offset calculado.

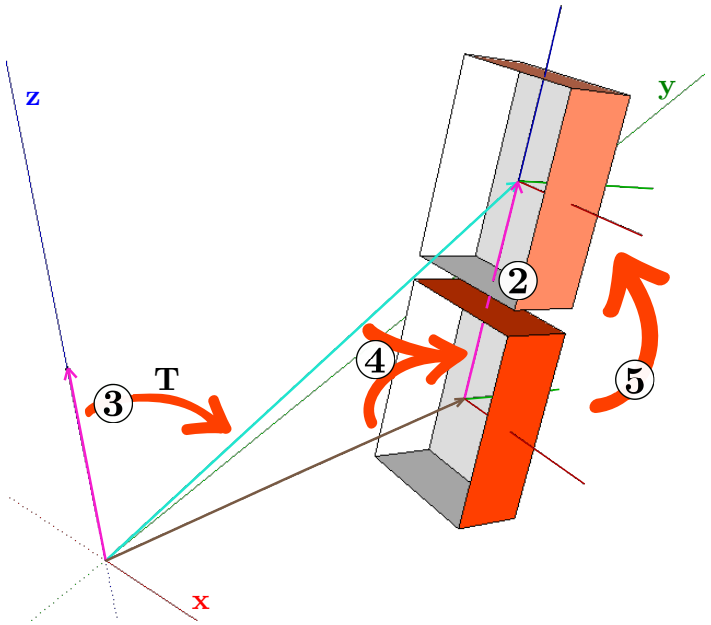

Figura 3.9: Traslación del instrumento a lo largo de su eje longitudinal

3.4.5.3. Izquierda/derecha

El movimiento izquierda/derecha es una rotación del objeto en torno al eje x (1,0,0) del sistema local de coordenadas de la esfera. Como dicho eje es paralelo al eje x del sistema global de coordenadas (tiene misma dirección y sentido), podrá realizarse la rotación alrededor de este último. Sin embargo, habrá que prestar especial atención al punto de apoyo de la rotación ya que, a diferencia de casos anteriores, será el centro de la esfera y no el centro del ortoedro.

En la Figura 3.10 se muestra la manera de proceder para realizar este tipo de rotaciones. El objeto en la representación se encuentra alineado en el plano *yz*, lo que supone una simplificación respecto al caso general. Esta simplificación no altera los cálculos realizados, ya que las componentes en x se mantienen iguales tras una rotación alrededor de dicho eje.

1. Rotar el ortoedro alrededor del eje x, con el punto de apoyo de la rotación situado en el centro del objeto. Ha de tenerse en consideración el sentido de giro:

   - Para una rotación en sentido antihorario, vista desde el alzado, el encoder proporciona un número de pulsos positivo.
   - Para un eje con sentido positivo en x, la función `rotateAboutGlobalAxisDeg` de Chai3d rota el instrumento en sentido antihorario, visto desde el alzado.
   - Como conclusión, el ángulo pasado como parámetro a la función ha de ser el obtenido a partir del número de pulsos del encoder.

   En la Figura 3.10 se realiza también la rotación de la esfera para ilustrar la necesidad de trasladar el objeto tras la rotación, de modo que el punto de anclaje del ortoedro a la esfera siga siendo el mismo.

2. Trasladar el ortoedro para devolver a su posición original el punto de anclaje con la esfera, que es, precisamente, el centro de la misma.

   a) El offset de la traslación resultará de restar la posición inicial de la esfera y la posición de la misma tras la rotación ilustrada en la Figura, estando ambas posiciones referenciadas al mismo sistema de coordenadas.

   b) En este caso, tomaremos como sistema de referencia el sistema de coordenadas local del ortoedro.

   c) Las coordenadas de la posición inicial de la esfera respecto al sistema elegido resultarán de restar la posición global de la esfera y la posición global del sistema elegido (esto es, la posición del ortoedro).

   d) Las coordenadas de la esfera tras la rotación se calcularán multiplicando el vector de posición inicial de la esfera (referido al sistema de coordenadas local del ortoedro) por la *matriz de rotación en torno al eje x*, R_x (ver Ecuación 3.1).

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sen(\alpha) \\ 0 & -sen(\alpha) & cos(\alpha) \end{bmatrix} \tag{3.1}$$

(Ramos et al., 2009)

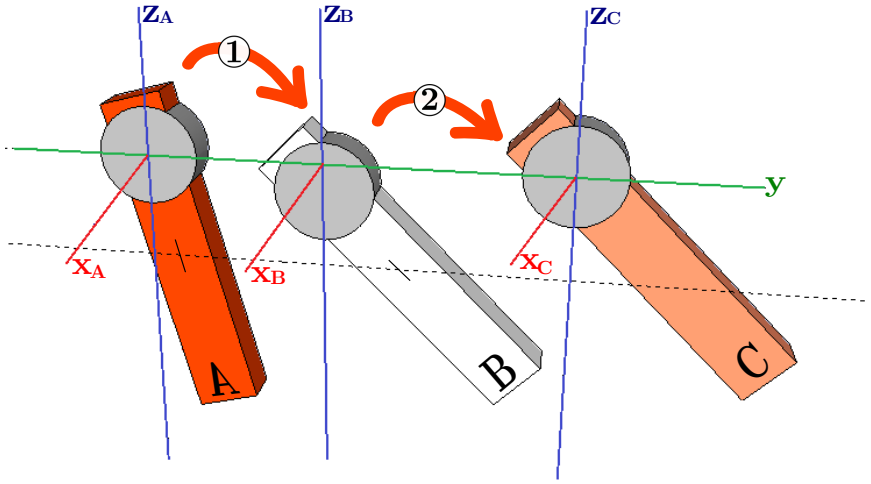

Figura 3.10: Rotación del ortoedro en torno al eje x del sistema de coordenadas local de la esfera.

3.4.5.4. Arriba/abajo

El movimiento arriba/abajo es una rotación del objeto en torno al eje y (0,1,0) del sistema local de coordenadas de la esfera. La manera de proceder es análoga a la vista en la Sección 3.4.5.3, con las siguientes salvedades:

- Distintas consideraciones respecto al sentido de giro.
  - Para una rotación en sentido antihorario, vista desde el perfil derecho, el encoder proporciona un número de pulsos negativo.
  - Para un eje con sentido positivo en y, la función `rotateAboutGlobalAxisDeg` de Chai3d rota el instrumento en sentido antihorario, visto desde el perfil derecho.
  - Como conclusión, el ángulo pasado como parámetro a la función ha de tener signo opuesto al obtenido a partir del número de pulsos del encoder.
- La *matriz de rotación en torno al eje y*, R_y, sigue la Ecuación 3.2).

$$R_y = \begin{bmatrix} cos(\alpha) & 0 & -sen(\alpha) \\ 0 & 1 & 0 \\ sen(\alpha) & 0 & cos(\alpha) \end{bmatrix} \tag{3.2}$$

(Ramos et al., 2009)

3.4.6. Funcionamiento del programa

Antes de lanzar el ejecutable generado a partir del código en C++, es necesario:

- Cargar en el Arduino Due el programa encargado de leer las señales de los encoders y enviar las tramas de datos por el puerto serie.
- Colocar los instrumentos tubulares perpendicularmente al plano del suelo.

Realizados estos pasos previos, ejecutamos el programa. Al comienzo, deben mantenerse los instrumentos unos segundos en la posición inicial de reposo mientras se espera a que el progama abra el puerto serie para la lectura de los datos y desde Arduino se reinicie el envío de dichos datos.

Una vez transcurrido este período de configuración inicial, pueden moverse libremente los instrumentos tubulares y seguir su movimiento por la pantalla del ordenador, tal y como muestra la Figura 3.11. Además, puede utilizarse el ratón para cambiar la posición de la cámara y obtener una nueva perspectiva de los instrumentos.

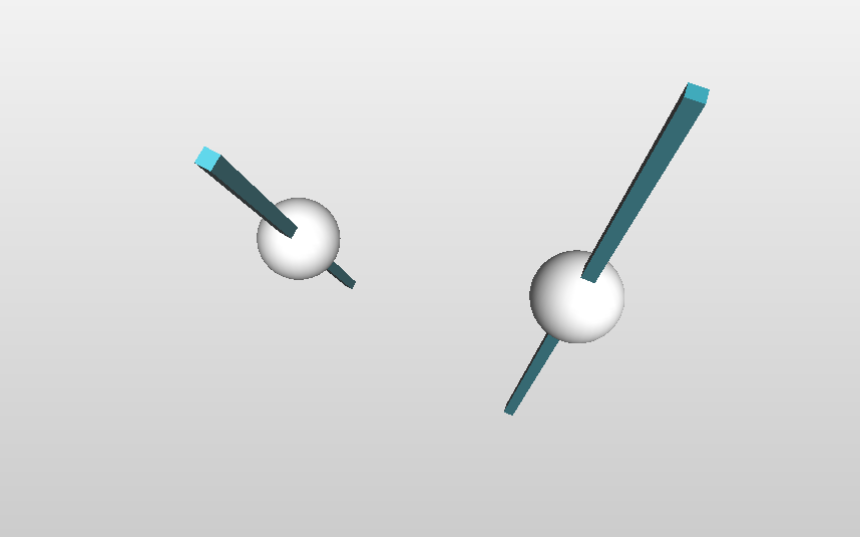

Figura 3.11: Captura de pantalla del programa de visualización de los instrumentos en tres dimensiones.

Capítulo 4

Conclusiones y líneas futuras

4.1. Conclusiones

En el presente Trabajo Fin de Grado han logrado subsanarse las dificultades e imprevistos, alcanzando el objetivo final de desarrollar dos programas de visualización. Ambos programas cumplen de manera correcta con su función principal, la representación bidimensional y tridimensional en tiempo real de los datos del entrenador laparoscópico, aun presentando un margen de mejora.

El cuello de botella del sistema global se encuentra en la decodificación de las tramas de datos. Este proceso está limitado por la velocidad del medio de transmisión (se decodifica a medida que se van recibiendo los bits) y por la cantidad de datos a transmitir, imponiendo periodos cercanos a los 40 ms. Haciendo uso de las herramientas adecuadas para la visualización y aprovechando las limitaciones humanas de la percepción visual, estas restricciones no han supuesto un impedimento para obtener gráficas animadas con tiempos de respuesta tolerables.

Partiendo de la caja entrenadora laparoscópica como eje vertebrador del trabajo y sin entrar en consideraciones sobre alternativas a esta, se ha intentado que el resto de los elementos utilizados fueran software y hardware libre y de bajo coste, de forma que el sistema global fuera fácilmente reproducible. Los subsistemas de visualización tridimensional y de adquisición y medida de las señales de los encoders sí han cumplido con dichas pretensiones al hacer uso de Chai3d y de la tarjeta y entorno de desarrollo de Arduino.

No ha sido así en el caso del programa de visualización de gráficas bidimensionales. Las limitaciones temporales del software libre planteado inicialmente como solución (CImg) han provocado la elección de Matlab como alternativa, que ofrece gran capacidad de cómputo pero es software propietario. No obstante, dado que las líneas futuras de mejora partirán de la solución de representación tridimensional, el uso de uno u otro tipo de software para las gráficas bidimensionales no se considera motivo de discusión.

4.2. Líneas futuras

Las líneas futuras del presente Trabajo Fin de Grado van orientadas a aumentar las funcionalidades y ámbitos de aplicación del simulador, así como a mejorar las características actuales del sistema. Cabe destacar las siguientes líneas de actuación:

1. **Mejorar la velocidad de transmisión y decodificación de los datos** para cumplir con los requisitos temporales exigidos por sistemas hápticos (f=1kHz).

   - Utilizar un protocolo y medio de transmisión alternativo que soporte mayores velocidades, como puede ser Ethernet.
   - Reducir la cantidad de datos a transmitir, alterando la longitud de la trama de datos:
     a) Asignar un orden y número de bits fijo a cada posición, de forma que no sea necesario el uso de identificadores.
     b) Ajustar el número de bits utilizados para un valor de posición al rango de valores posibles que esta posición pueda tomar.
     c) Ajustar la resolución de los valores de posición hasta el mínimo aceptable, de manera que se reduzca el rango posible de valores que pueden tomar.

2. **Mejorar la precisión del instrumento**, abarcando las siguientes modificaciones:

   - Fabricar una estructura o acoplar elementos mecánicos que permitan fijar una posición inicial del instrumento con exactitud.
   - Tomar de manera exacta las medidas de la caja entrenadora: dimensiones de los instrumentos, radios de rotación, disposición de los elementos...
   - Arreglar deficiencias mecánicas de la estructura: falta de estabilidad de la base, incorrecto arrollamiento de hilos metálicos, sistema de apertura y cierre de las pinzas laparoscópicas...

3. **Desarrollar un sistema completo de simulación virtual**, llevando a cabo las siguientes acciones:

   - Elaborar un modelo tridimensional del instrumento en detalle, buscando un compromiso entre el realismo del modelo y la capacidad de cómputo necesaria para procesar los movimientos.
   - Añadir funcionalidad a las pinzas laparoscópicas en el entorno simulado, incluyendo sensores que permitan obtener los datos necesarios para reproducir fielmente los movimientos de apertura y cierre correspondientes.
   - Desarrollar un subsistema de realimentación háptica, que permita la interacción del usuario con el entorno virtual de simulación mediante la percepción táctil.
   - Implementar el entorno virtual de actuación, incluyendo varios modelos de regiones anatómicas susceptibles de requerir una operación mediante laparoscopia. Debe buscarse el equilibrio entre el realismo de los modelos anatómicos y la complejidad del cómputo necesario para procesar las fuerzas de realimentación háptica correspondientes al modelo.

Bibliografía

Altman, Y. M. (2015). *Accelerating MATLAB Performance: 1001 tips to speed up MATLAB programs*. CRC Press, 6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL.

Arduino (2016a). Arduino Due: Overview. `https://www.arduino.cc/en/Main/ArduinoBoardDue`. [Online; accedido 11/7/2016].

Arduino (2016b). Getting started with the Arduino Due:Serial ports on the Due. `https://www.arduino.cc/en/Guide/ArduinoDue#toc3`. [Online; accedido 11/7/2016].

Arduino (2016c). Reference: attachInterrupt(). `https://www.arduino.cc/en/Reference/AttachInterrupt`. [Online; accedido 11/7/2016].

Arduino (2016d). Serial:println(). `https://www.arduino.cc/en/Serial/Println`. [Online; accedido 11/7/2016].

ASCRS (2016). Cirugía Laparoscópica. `www.fascrs.org/cirugia-laparoscopica`. [Online; accedido 11/7/2016].

Bakaus, P. (2014). The Illusion of Motion. `https://paulbakaus.com/tutorials/performance/the-illusion-of-motion/`. [Online; accedido 11/7/2016].

Castañeda Marín, R. (2016). Aplicación de matrices de transformación en el control de posición cinemático de un robot articulado de tres grados de libertad. `http://forja.uji.es/docman/view.php/43/46/PonenciaRobotica.pdf`. [Online; accedido 13/7/2016].

Davis, J. et al. (2015). Humans perceive flicker artifacts at 500 hz. *Nature*, 5(7861).

Die.net (2016). Linux man page. `http://linux.die.net/man/8/setserial`. [Online; accedido 11/7/2016].

Eltra, SILGE Electrónica S.A. (2008). Encoder incremental. Descripción general. `http://facultad.bayamon.inter.edu/arincon/encoderincrementales.pdf`. [Online; accedido 11/7/2016].

Gagalowicz, A. y Philips, W. (2005). *Computer Analysis of Images and Patterns. 11th International Conference, CAIP*. Springer, Versailles, France.

García Berro, M. y Toribio, C. (2004). El futuro de la cirugía mínimamente invasiva: Tendencias tecnológicas a medio y largo plazo. Technical report, Fundación OPTI y FENIN.

Gurley Precision Instruments (2002). Understanding quadrature. `www.gurley.com/Encoders/Understanding_Quadrature.pdf`. [Online; accedido 11/7/2016].

Kuchenbecker, K. J. et al. (2006). Improving contact realism through event-based haptic feedback. *IEEE Transactions on visualization and computer graphics*, 12(2):219–230.

Laparoscópica (2016). Trócares laparoscópicos. `www.laparoscopica.es/instrumentos/trocar`. [Online; accedido 11/7/2016].

Lewis, J. (2012). *Essential Cinema: An Introduction to Film Analysis*. Wadsworth, 20 Channel Center Street, Boston, MA 02210, USA.

Linurs (2016). Serial port: Networking. `http://www.linurs.org/linux/SerialPort.html`. [Online; accedido 11/7/2016].

MAPFRE Salud (2016). Pruebas Diagnósticas. `www.mapfre.es/salud/es/cinformativo/laparoscopia.shtml`. [Online; accedido 11/7/2016].

Martínez Valdéz, A. et al. (2016). Gui multi-resolución parte 3: World space - sistema de coords del mundo. `https://davidtheory.wordpress.com/category/unity3d/`. [Online; accedido 13/7/2016].

MathWorks Support Team (2016). Matlab answers. `https://es.mathworks.com/matlabcentral/answers/95024-why-is-my-serial-port-not-recognized-with-matlab-on-linux-or-solaris`. [Online; accedido 11/7/2016].

Negus, C. y Weeks, T. (2001). *Linux Troubleshooting Bible*. Wiley Publishing, Inc., 10475 Crosspoint Boulevard, Indianapolis, IN 46256.

Ni, Z. et al. (2014). *Haptic Feedback Teleoperation of Optical Tweezers*. ISTE Ltd y John Wiley & Sons, Inc., 23-27 St George's Road, London SW19 4EU, UK.

Piéron, H. (1990). *Vocabulario Akal de Psicología*. Ediciones Akal, S.A, Los Berrocales del Jarama, Apartado 400, Torrejón de Ardoz, Madrid, España.

Poynton, C. (2012). *Digital Video and HD: Algorithms and Interfaces*. Morgan Kaufmann, Elsevier, 225 Wyman Street, Waltham MA 02451 USA.

Ramos, R. et al. (2009). Tema II: Transformaciones lineales en 3D. `http://di002.edv.uniovi.es/~rr/Tema2.pdf`. [Online; accedido 14/7/2016].

Sánchez-Margallo, Juan A. et al. (2014). Utilidad de un sistema de seguimiento óptico de instrumental en cirugía laparoscópica para evaluación de destrezas motoras. *Cirugía Española*, 92(6):421–428.

Sánchez-Peralta, L.F. et al. (2010). Evaluación objetiva de destrezas laparoscópicas básicas a partir de métricas obtenidas con el simulador virtual sinergia. In *XXVIII Congreso Anual de la Sociedad Española de Ingeniería Biomédica*.

Tarco-Delgado, R. et al. (2007). Entrenamiento laparoscópico en un modelo para prácticas domiciliarias. *Revista Peruana de Urología*, (16):11–14.

The Linux Documentation Project (2011). Serial Port HowTo. `http://www.tldp.org/HOWTO/Serial-HOWTO-12.html`. [Online; accedido 11/7/2016].

Venegas Requena, J. (2009). Encoders. `http://ramos.elo.utfsm.cl/~elo212/docs/Encoders-jvr-v01.pdf`. [Online; accedido 11/7/2016].