CONTROL PID PARA LOS ROBOTS HIBOU Y LAPBOT

ANDRÉS FELIPE NÚÑEZ CONCHA

HERNÁN FELIPE SOLARTE VÉLEZ

Director: Dr. (PhD)

OSCAR ANDRÉS VIVAS ALBÁN

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Electrónica, Instrumentación y Control

Ingeniería en Automática Industrial

Popayán, Febrero de 2016

CONTROL PID PARA LOS ROBOTS HIBOU Y LAPBOT

ANDRÉS FELIPE NÚÑEZ CONCHA

HERNÁN FELIPE SOLARTE VÉLEZ

Monografía presentada como requisito

parcial para optar por el título de

Ingeniero en Automática Industrial

Director: Dr. (PhD)

OSCAR ANDRÉS VIVAS ALBÁN

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Electrónica, Instrumentación y Control

Ingeniería en Automática Industrial

Popayán, Febrero de 2016

Nota de Aceptación: _____

_____

_____

_____

_____

Director: _____

Oscar Andrés Vivas Albán

_____

Firma del Jurado

_____

Firma del Jurado

Popayán, Febrero de 2016

Agradecimientos

A nuestras familias que siempre estuvieron ahí para apoyarnos en todo momento.
A los profesores que contribuyeron con sus enseñanzas y experiencias, para hacernos crecer personal y profesionalmente durante este largo periodo de carrera.
A nuestros compañeros de estudio por sus consejos, apoyo y su invaluable amistad.
A la sección estudiantil ISA Unicauca por darnos a conocer un panorama más profesional de nuestra carrera y permitirnos compartir experiencias inolvidables.
A nuestro director de tesis Andrés Vivas que nos brindó su colaboración en el ámbito intelectual y administrativo, para la realización de este proyecto.
A la Universidad del Cauca por su contribución administrativa y económica.
A Popayán por acogernos durante 6 años llenos de alegría y grandes logros.

Resumen

En la Universidad del Cauca, el grupo de I+D en Automática Industrial cuenta con dos robots construidos físicamente (Lapbot para cirugía laparoscópica y el robot porta endoscopio Hibou). La construcción de estos robots se llevó a cabo tras varios años de investigación y diversas tesis de pregrado y maestría en la Universidad del Cauca. Estos robots tienen una estructura mecánica bien diseñada, sin embargo existe una obsolescencia tecnológica en el *software* de control, simulación y el *hardware* de adquisición de datos.

En este proyecto se realiza el diseño e implementación de controladores PID paralelos para los robots con el fin de verificar su movilidad en un espacio operacional manteniendo un punto fijo en el espacio (trocar), y así validar el sistema teniendo como parámetro el error cartesiano.

Para realizar el control, adquisición de datos y comunicación serial con el computador se realiza el diseño de una tarjeta electrónica, la cual cuenta con un Arduino Mega2560.

Los controladores PID son simulados en Matlab para verificar su desempeño frente a consignas geométricas. Los sensores y motores de los robots son caracterizados realizando modificaciones al *hardware* y configuraciones para mejorar su rendimiento y se implementan y sintonizan los controladores PID paralelos en la tarjeta electrónica.

El software *RELAS Robotic Environment for Laparoscopic Surgery"* permite visualizar los robots en un entorno tridimensional realizando movimientos generados a partir de consignas geométricas o por medio del *joystick*, los robots se mueven mediante los datos realimentados de la tarjeta electrónica y se dispuso de ventanas gráficas para observar los errores articulares y cartesianos.

Se concluye que el controlador PID sigue siendo una alternativa funcional para el control de robots, este tuvo buen desempeño en el seguimiento de posiciones, conllevando a que el primer prototipo de los robots Hibou y Lapbot cumplan el objetivo de mostrar su capacidad de realizar movimientos en un entorno quirúrgico bajo ciertas especificaciones.

Abstract

At the University of Cauca, the research and development group of Industrial Automation has two robots physically built (Lapbot for laparoscopic surgery and robotic endoscope holder Hibou). The construction of these robots is conducted after several years of research and several undergraduate and master's thesis at the University of Cauca. These robots have a well-designed mechanical structure, but there is a technological obsolescence in the control software, simulation and data acquisition hardware.

In this project the design and implementation of a parallel PID controller for the robots in order to verify their mobility in an operating room while maintaining a fixed point in the operational space, and thus validate the system taking as performed parameter the Cartesian error .

For controling, data acquisition and serial communication with the computer, the design of an electronic board has been made having on it an Arduino Mega2560 .

The PID controller is simulated in Matlab to verify its performance in tracking geometric trajectories. Sensors and motors'robots are characterized making changes to hardware and configurations to improve performance. On the electronic board a parallel PID controller is implemented and tuned.

The software RELAS *Robotic Environment for Laparoscopic Surgery* allows viewing of robots in three-dimensional environment by performing movements generated from geometric trajectories or through a joystick, robots move through the data feedback from the control board and additional observing graphic windows to analyze the joint and cartesian errors.

Concluding that the PID controller is still a functional alternative for the control of robots, this had a great performance in tracking of positions, leading the first prototype of the Hibou and Lapbot robots to reach its target of showing their ability to perform movements in a surgical environment under certain specifications.

Contenido

Índice de Figuras

Índice de Tablas

Introducción

En los últimos años los avances tecnológicos y mejoras en conocimiento en muchos campos de la ciencia han transformado el mundo que nos rodea. La tecnología aplicada en el campo de la medicina ha cambiado sustancialmente permitiendo innovar la forma de prestar y suplir asistencia médica a las personas.

En los años 80 se dio inicio al desarrollo de técnicas de cirugía mini-invasiva, eliminando la necesidad de que el cirujano introdujera sus manos en el cuerpo del paciente[1]. Sin embargo este procedimiento agregó nuevos retos debido a la pobre realimentación de sensaciones al tocar los órganos internos, la pérdida de visión en tres dimensiones y la poca movilidad en el espacio operatorio.

Desde que Erich realizó por primera vez una colecistectomía por laparoscopia el 12 de septiembre de 1985 se revolucionó la cirugía en general, desde ese momento hasta la actualidad se tiende a practicar métodos poco invasivos facilitados por procedimientos como la laparoscopia evitando grandes heridas al paciente [2].

En ese mismo año Kwoh [3], realizó un procedimiento neuroquirúrgico con un robot Puma 560 de tipo industrial. Kwoh lo usó para sostener un accesorio junto a la cabeza de su paciente y realizar la biopsia.

En los años posteriores la robótica médica ha dirigido sus desarrollos a la asistencia de pacientes y médicos. En la primera se tienen dispositivos orientados hacia la rehabilitación de pacientes y a asistir a personas discapacitadas o en la tercera edad (prótesis, electro estimulación, asistentes personales, etc). En cuanto a los médicos, se encuentran los robots diseñados para cirugía, exploración, diagnóstico y terapia [4] .

El robot quirúrgico ampliamente utilizado hasta el día de hoy es Da Vinci, el cual cuenta con una de las más altas tecnologías al servicio de los cirujanos con controles ergonométricos avanzados que permiten manipular a través de una visión de alta resolución en tres dimensiones, instrumentos diseñados con grados de movimiento que imitan y superan al de la muñeca y mano humana. De esta manera, el cirujano dispone de herramientas que superan los beneficios de la cirugía laparoscópica tradicional [5].

Las principales investigaciones y desarrollos sobre robótica se han realizado en los Estados Unidos y países de Europa en general, para países en desarrollo la adquisición de estas tecnologías se hace difícil por su elevado costo (El robot Da Vinci puede costar aproximadamente 1.5 millones de dólares[6]).

En Colombia, existen muchas universidades e instituciones realizando proyectos en el campo de la robótica médica. Muchos de estos proyectos se han centrado en el desarrollo de prótesis de extremidades y rehabilitación de miembros superiores e inferiores, además, de generar ambientes tridimensionales virtuales para entrenamiento quirúrgico. Sin embargo los proyectos de robots en ambientes quirúrgicos son pocos y en general no están terminados, pues solo han llegado a la etapa de simulación.

En la Universidad del Cauca el grupo de Automática Industrial ha desarrollado diversos proyectos en el campo de la robótica médica orientados a cirugía laparoscópica. Dos tesis de pregrado culminaron satisfactoriamente con la construcción de los primeros prototipos reales del robot porta endoscopio Hibou y del robot quirúrgico Lapbot.

La movilidad de los robots en el espacio cartesiano para el posicionamiento del robot y su respeto en el paso por el trocar, así como los movimientos quirúrgicos que deben ser realizados por el cirujano representan el mayor desafío cuando se cuenta con robots para ser llevados al campo quirúrgico.

Debido a esto y por su importancia, el presente proyecto pretende el diseño e implementación de un controlador PID a cada uno de estos robots, así como también un nuevo diseño en el *hardware* para comunicación, adquisición y transmisión de señales.

El documento se encuentra dividido en seis capítulos. El primer capítulo se presentan algunos asistentes robóticos usados en el pasado y la actualidad y tipos de controladores, además de una introducción a los simuladores quirúrgicos y los robots quirúrgicos desarrollados en la Universidad del Cauca. El segundo capítulo se presenta de forma breve las características de los robots y el diseño de la nueva tarjeta de adquisición de datos con sus componentes electrónicos. En el tercer capítulo se diseña y simula el controlador PID y se realiza la implementación del controlador en la nueva tarjeta de adquisición de datos. El cuarto capítulo presenta lo referente al software *RELAS* para manipulación de los robots en un ambiente 3D, su forma de desarrollo y consignas geométricas realizadas. El quinto capítulo cinco se presentan un conjunto de recomendaciones planteadas como resultado de un análisis a todo el proyecto. Finalmente el sexto capítulo quedan consignadas las conclusiones del proyecto y los trabajos futuros.

Capítulo 1

Robótica y Medicina

1.1. Robótica quirúrgica

1.1.1. Antecedentes

Desde hace algunas décadas se dio inicio al desarrollo de los primeros robots, estos pasaron de máquinas rudimentarias a robots con tareas complejas en distintos campos de la industria, sustituyendo a operadores donde su integridad física estaba en peligro. Desde hace tiempo los robots han sido empleados en el campo de la medicina, y en la actualidad en una gran variedad de procedimientos quirúrgicos.

Si bien la laparoscopia revolucionó el campo de la cirugía, hoy en día la robótica está revolucionando el campo de la cirugía *"minimally invasive surgery"* MIS permitiendo que muchos procedimientos de alta exigencia técnica se vean asistidos por esta tecnología, contribuyendo a mejorar el desempeño quirúrgico, acotar las curvas de aprendizaje en determinados procedimientos, ampliar la utilidad y la precisión en un procedimiento de laparoscopia [7].

A su vez, la tecnología robótica ha dado importantes aportes al campo laparoscópico, por ejemplo:

- Visión de profundidad tridimensional.
- Movimiento multigrado de libertad que imita y mejora la articulación de la mano del cirujano.
- Posibilidad de tutoría y asistencia a distancia de procedimientos quirúrgicos complejos

Figura 1.1: Robot Puma 560.
Fuente: Tomada de [8].

Los robots destinados a procedimientos médicos han evolucionado hasta incluir las condiciones físicas y de seguridad necesarias en un ambiente quirúrgico, además su implementación ha permitido a los cirujanos aumentar la precisión y eliminar posibles errores debidos al temblor, cansancio o movimientos involuntarios del cirujano.

Todo teniendo como base previa un entrenamiento basado en simulación, el cual familiariza al cirujano con los procedimientos quirúrgicos sin correr el riesgo de cometer un error que afecte al paciente.

En 1985, Kwoh utilizó el primer robot no laparoscópico modificando un robot industrial estándar, el robot Puma 560, para realizar una biopsia de cerebro con gran precisión. Kwoh lo usó para sostener un accesorio junto a la cabeza de su paciente y realizar la biopsia sin temor a que las agujas pudieran insertarse en un lugar indeseado[3].

Tres años después, Davies realizó una resección transuretral de la próstata usando el robot Puma 560, este sistema llevó al desarrollo de *PROBOT*, un robot diseñado específicamente para la resección transuretral de la próstata. Mientras tanto *"Integrated Surgical Supplies Ltd"*, estaba desarrollando *ROBODOC*, un sistema robótico diseñado para mecanizar el fémur con una mayor precisión en una cirugía de reemplazo de cadera [9].

A partir de estos acontecimientos, las investigaciones y desarrollos de robots para este tipo de procedimientos crecieron a un ritmo semejante a las mejoras tecnológicas y la habilidad de los cirujanos [9].

1.1.2. Robots quirúrgicos comerciales

AESOP

AESOP *"Automated Endoscopic System for Optimal Positioning"* el cual fue usado clínicamente en 1993 y comercializado en 1994 como el primer robot quirúrgico aprobado por la FDA *"US Food and Drug Administration"* [10].

AESOP fue fabricado por *"Computer Motion Incorporated"* USA. Esta compañía fue inicialmente fundada con una concesión para investigación de la NASA, con el fin de desarrollar un brazo robótico para el programa espacial de los Estados Unidos. Posteriormente este brazo fue modificado para sostener un laparoscopio y reemplazar al asistente encargado en esta tarea [11].

Figura 1.2: Colectomía robótica con AESOP.
Fuente: Tomada de [11].

En las primeras versiones de AESOP, el cirujano controlaba el brazo robótico ya sea manualmente o a distancia a través de un pedal o un control manual, estos podían programarse en tres posiciones preestablecidas para tareas repetitivas; versiones más recientes obedecen a comandos de voz. Fue catalogado como una herramienta importante en la cirugía por la estabilidad que proporcionaba a las imágenes durante la operación del paciente [11, 12]. Luego sufrió un rediseño para dar nacimiento al sistema ZEUS, uno de los sistemas quirúrgicos más emblemáticos en la cirugía laparoscópica [9].

ZEUS

El sistema quirúrgico robótico ZEUS, diseñado para asistencia en la cirugía, fue producido por la empresa de robótica *"Computer Motion"*. En este sistema, el robot AESOP sujeta la cámara y dos unidades adicionales semejantes a este han sido modificadas para sostener los instrumentos quirúrgicos, esto permite ser un sistema de tres brazos robóticos independientes que son acoplados a la mesa de la sala de operaciones [11].

Figura 1.3: Sistema quirúrgico ZEUS.
Fuente: Tomada de [8].

Con este robot se realizó la primera operación tele-operada desde un centro de mando en New York, en donde un cirujano realizó una colecistectomía completa a un paciente en Strasbourg, Francia en el año 2001 [13].

DA VINCI

El sistema quirúrgico Da Vinci [14] es una sofisticada plataforma robótica desarrollada en 1983 orientada a ampliar las capacidades del cirujano. El primer diseño llamado estándar fue creado en 1999 por la compañía *"Intuitive Surgical"*, y aprobado por la FDA en el año 2000.

Hasta el 2013 existían al menos 3.000 sistemas quirúrgicos Da Vinci instalados en el mundo, entre todos ellos han realizado más de 300.000 cirugías. La versión actual Da Vinci Xi, es un carro móvil que cuenta con cuatro brazos, uno de ellos es portador de una cámara tridimensional de alta resolución, y los tres restantes sostienen los instrumentos quirúrgicos [14, 15].

Figura 1.4: El sistema quirúrgico telerobótico Da Vinci.
Fuente: Tomada de [11].

Da Vinci posee un sistema de control de vibraciones para la motricidad del cirujano, reduciendo al mínimo los movimientos involuntarios, además de permitir una escalabilidad de los movimientos del cirujano hasta 5 veces, de manera que los instrumentos se puedan mover de forma más precisa que la mano humana [14].

TELELAP ALF-X

TELELAP ALF-X, *"Advanced Laparoscopy through Force Reflection"* nació a partir de un estudio de viabilidad realizado como parte de una tesis sobre robótica en el Politécnico de Milán, Italia. TELELAP ALF-X ha sido diseñado para ser utilizado en una variedad de procedimientos quirúrgicos incluyendo la ginecología, urología, cirugía general y cirugía torácica, permitiendo al cirujano realizar procedimientos quirúrgicos a través de la estación de control. TELELAP ALF-X brinda al cirujano la posibilidad de maniobrar como si estuviera utilizando los instrumentos laparoscópicos habituales que, naturalmente, aceleran la familiarización y formación inicial. Esta tecnología realiza los movimientos de la cirugía laparoscópica tradicional, aumentando su eficacia, precisión y calidad [16].

Figura 1.5: Entorno quirúrgico TELELAP ALF-X.
Fuente: Tomada de [16].

Este sistema robotizado realiza retroalimentación háptica innovadora a través de la consola *"cockpit"* que permite que el cirujano pueda "sentir"la fuerza empleada a través de los instrumentos y la resistencia natural de los tejidos, la fuerza de retroalimentación es particularmente útil, por ejemplo, la sensación de la aguja que pasa a través del tejido y el tirón de la puntada son de gran valor y mejoran la calidad de la sutura.

Su sofisticada forma de determinar el punto de menor resistencia dentro de un trocar hace que TELELAP interactúe suavemente con la pared abdominal y el tejido alrededor del sitio de la incisión con el objetivo de evitar el exceso de presión y fricción.

SOFIE

La Universidad Tecnológica de Eindhoven, Holanda ha creado un prototipo de robot quirúrgico, *"Surgeon's Operating Force-feedback Interface Eindhoven"* Sofie[17], el cual ha sido el primer diseño de robots quirúrgicos que proporciona retroalimentación haptica. Este sistema podría ser especialmente útil para tareas tales como hacer suturas, ya que brinda a los cirujanos una mejor idea de la fuerza con que están tirando del hilo.

Figura 1.6: Robot quirúrgico Sofie.
Fuente: Tomada de [17].

Sofie es más compacta que la mayoría de los robots quirúrgicos, y se monta en la mesa de operaciones en lugar del suelo, esto significa que cuando la mesa se inclina o se mueve dentro de la sala, Sofie se moverá con ella, así que no necesitará de reajustes.

Este sistema de retroalimentación de fuerza ha sido patentado, y se espera que la comercialización de Sofie sea posible en los próximos años.

MIROSURGE

El escenario para telecirugía MiroSurge[18] se ha desarrollado en MiroLab, un laboratorio de asistencia robótica avanzada en el *"Robotic and mechatronic center"* DLR, Alemania. MiroSurge incluye una consola de control, así como un teleoperador que consta de 3 robots quirúrgicos tipo Miro, los cuales son brazos robóticos para aplicaciones quirúrgicas desarrollados por el DLR. Dos Miros llevan instrumentos quirúrgicos equipados con sensores de fuerza y torque para capturar las fuerzas de reacción con el tejido manipulado. El otro Miro automáticamente guía un laparoscopio con una cámara de vídeo estereoscópica. Tanto el flujo de vídeo y las fuerzas medidas se muestran al cirujano en la

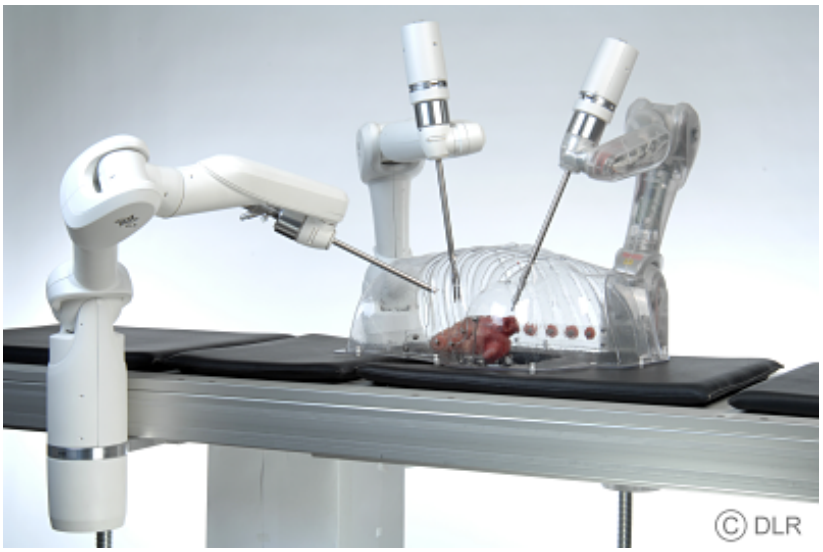

Figura 1.7: Escenario para telecirugía MiroSurge.
Fuente: Tomada de [18].

consola principal. Así los cirujanos no se limitan a ver, sino también pueden sentir lo que están haciendo.

La ambición de MiroSurge es apoyar cirugías de corazón. Las características de rendimiento de los Miros están diseñados para seguir un movimiento del corazón mientras late, mostrando un flujo de video estabilizado por el seguimiento en tiempo real.

Las siete articulaciones de torque controlado de los robots Miro permiten crear una sala de operaciones más flexibles evitando colisiones con otros robots o equipo de la sala de operaciones.

1.1.3. Controladores de manipuladores robóticos

Un sistema robótico está constituido por una estructura mecánica, un controlador, elementos de control y potencia, que en conjunto permiten la movilidad del sistema. El controlador es el encargado de la correcta movilidad del robot, protegiendo de movimientos bruscos o fuera de los topes mecánicos que alteren o dañen la mecánica del robot [19].

Control PID

El control PID es una de las estrategias de control más utilizadas para el control de robots industriales y quirurgicos, su sencillez y su amplia gama de aplicaciones lo posicionan en el mercado como la primera opción al implementar un sistema de control. La funcionalidad y buen desempeño incluso desconociendo el modelo matemático de la planta, hace

de este la mejor opción para el diseño del controlador para los robots, además, el control PID cuenta con buena respuesta para el seguimiento de trayectorias y sus parámetros se pueden ajustar fácilmente y por separado [20, 21], a continuación se citan algunas de las multiples aplicaciones e implementaciones de este controlador.

En *"Implementación de una estación de trabajo mediante un robot serial de 3 grados de libertad"* [22] se diseña e implementan las interfaces *hardware* y *software*, el algoritmo de control PID Desacoplado, así como la interface de usuario que permita comandar movimientos geométricos en 2D de un robot serial planar de 3 grados de libertad.

En *"Force Signal Tuning for a Surgical Robotic Arm Using PID Controller"*[23], se sintoniza un controlador PID para mejorar y ajustar la fuerza de salida del brazo de dos eslabones.
Otras aplicaciones de controladores PID se pueden ver en [24, 25][26].

Control por par calculado

El controlador por par calculado CTC es un controlador no lineal de gran alcance que se utiliza ampliamente en el control de robots manipuladores. Este controlador funciona muy bien cuando se conocen todos los parámetros dinámicos y físicos, pero cuando los parámetros dinámicos tienen variaciones el controlador no tiene un rendimiento aceptable en el seguimiento de las trayectorias. En la práctica, la mayoría de los parámetros de los sistemas físicos son la variante desconocida o tiempo[27].
En *"Towards Sophisticated Control of Robotic Manipulators: An Experimental Study on a Pseudo-Industrial Arm"*[28] se diseñó, simuló e implementó el *hardware* de dos estrategias de control: el control por par calculado CTC y el control de estructura variable en un manipulador de pseudo-industrial con seis grados de libertad.
En *"Control basado en modelo para robots paralelos con sensorización redundante"*[29] donde se presenta una nueva estrategia de control CTC extendido estable, validando su funcionamiento sobre un robot Delta utilizando el *software* multibody ADAMS.

Control Predictivo

Muchos trabajos han mostrado que es una opción interesante para el manejo de diversos procesos, al realizarse una optimización del pronóstico del comportamiento del sistema. Este control tiene buena respuesta en términos de rapidez, rechazo de perturbaciones y respuesta frente a errores en los parámetros de la planta. Sin embargo, aunque un significativo número de aplicaciones industriales pueden ser encontradas en procesos químicos o de fabricación de alimentos, donde las dinámicas a tratar son relativamente lentas, pocos

resultados pueden encontrarse en el control de procesos no lineales y con alta dinámica, como es el caso de los robots manipuladores tipo *SCARA* o destinados a ambientes quirúrgicos, donde además se manejan órdenes de pocos milisegundos [30].
Una aplicación de este tipo de controlador se puede apreciar en *"Application of predictive control techniques within parallel robot"* [31] donde se utiliza como estrategia un control predictivo generalizado GPC en la forma RST, en el cual se considera el modelo dinámico linealizado del robot *"Orthoglide"*. Los controladores se utilizan para realizar seguimiento de trayectoria, donde se simula teniendo en cuenta incertidumbres de *OrthoGlide* relacionados con parámetros geométricos y dinámicos, ruido sensores y fricciones, con dos trayectorias diferentes y se compara el controlador GPC, RGPC y el clásico CTC.

1.2. Avances tecnológicos en la Universidad del Cauca acerca de cirugía laparoscópica robotizada

1.2.1. Lapbot

La Universidad del Cauca realizó una investigación en la cual se toman como base los diferentes robots comerciales y no comerciales diseñados para cirugía mínimamente invasiva en el mundo con el fin de especificar una serie de requerimientos que debe tener un manipulador de este tipo. Como resultado se realizó el diseño de la arquitectura del robot llamado Lapbot (ver figura 1.8)[32], el modelo cinemático y dinámico, y el modelo de una restricción matemática que se impone al robot para que pase por una incisión fija realizada en la cavidad abdominal del paciente.

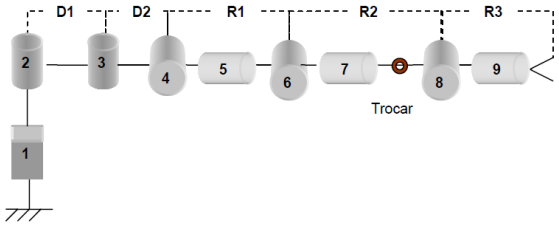

Figura 1.8: Estructura cinemática del robot Lapbot.
Fuente: Tomada de [32].

Los modelos del robot fueron simulados en un ambiente tridimensional donde se logra una comprobación visual tridimensional de la funcionabilidad de Lapbot con trayectorias de una colecistectomía (extracción de una vesícula biliar enferma) [33]. Un controlador

CTC propuesto y diseñado fue probado usando diversas trayectorias cartesianas típicas en una intervención quirúrgica, los errores de seguimiento obtenidos fueron bastante buenos, corroborándose la potencialidad del robot para realizar operaciones de laparoscopia [34].

1.2.2. Hibou

Posteriormente se diseñó un robot porta endoscopio, con una estructura cinemática distinta, que lleva por nombre Hibou. [35].

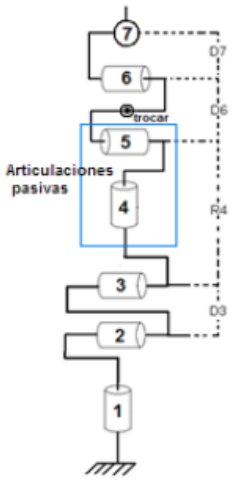

Figura 1.9: Estructura cinemática del robot Hibou.
Fuente: Tomada de [35].

Hibou (ver figura 1.9), ha sido diseñado con 7 grados de libertad, modelado a nivel geométrico, dinámico, y simulado en un ambiente tridimensional. Este robot permite seguir con precisión trayectorias que se necesitan en procedimientos laparoscópicos, manteniendo un punto fijo en el espacio para la rotación del efector final que se introduce al paciente, es decir, respetando el paso por el trocar [35].

1.2.3. PA-10

Como hecho reciente, se ha realizado un estudio sobre el uso de robots industriales como asistentes en operaciones de laparoscopia, abordando de manera particular el problema del paso por el orificio abdominal (trocar). Se estudiaron dos robots industriales (Puma de Unimation y PA-10 de Mitsubishi).

Los resultados mostraron que al poseer un grado de libertad adicional en el codo, el robot PA-10 (ver figura 1.10), cumplía con las condiciones impuestas al problema: realizar

una consigna quirúrgica al interior del abdomen del paciente respetando al mismo tiempo el paso por el trocar [36].

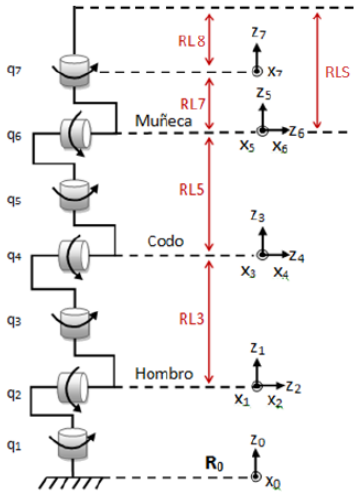

Figura 1.10: Estructura cinemática del robot PA-10.
Fuente: Tomada de [37].

Derivado de este estudio, se llevó a cabo el diseño e implementación de una solución *software* que verificara el comportamiento del robot PA-10 en aplicaciones quirúrgicas teniendo como prioridad el respeto del paso por el trocar en la cavidad abdominal. Como trabajo futuro se cuenta con las piezas CAD del diseño del PA-10 con miras a su construcción[37].

1.3. Simuladores y entrenadores quirúrgicos

El entrenamiento es un elemento fundamental cuando se piensa en mejorar el rendimiento sin importar que tan compleja o especializada sea la actividad. En el campo de la medicina, cuando se realiza una cirugía, cualquier fallo debido a la inexperiencia del cirujano puede resultar con consecuencias negativas e incluso fatales para el paciente. En la actualidad el entrenamiento de los cirujanos se lleva a cabo mediante la utilización de cadáveres, maniquíes, animales vivos o en intervenciones quirúrgicas reales bajo la supervisión de un experto [38].

La evolución actual de los ordenadores, más concretamente el *hardware* y el *software* gráfico, han permitido el desarrollo de entornos virtuales. simuladores de diversos tipos y en diferentes campos de la ciencias. Para el caso de la cirugía se ha aportado de forma significativa simuladores quirúrgicos, los cuales permiten:

- Proveer de experiencia al cirujano con una mayor variedad de patologías y complicaciones.

- Permitir la posibilidad de repetir los procedimientos quirúrgicos tantas veces como sea necesario hasta su correcto aprendizaje.

- Permitir revisualizar los procedimientos realizados con el objetivo de poder estudiar sus ventajas e incluso mejorarlos mediante la utilización de técnicas diferentes a las empleadas.

1.3.1. LapMentor

Entre las diferentes soluciones comerciales desarrolladas para la simulación de cirugía laparoscópica se destaca LapMentor[39], el cual proporciona el más alto nivel de formación quirúrgica, ofreciendo una solución completa en todos los niveles y disciplinas incluyendo procedimientos quirúrgicos en ginecología, urología y cirugía general. LapMentor cuenta con la opinión de expertos en cirugía, sociedades médicas y educadores con el fin de mejorar continuamente y brindar un producto de mayor impacto .

Figura 1.11: Simulador LapMentor.
Fuente: Tomada de [39].

1.3.2. CAE ProMIS Laparoscopic Simulator

CAE ProMIS Laparoscopic Simulator [40], permite a los cirujanos interactuar con modelos físicos y virtuales en la misma unidad al mismo tiempo mientras proporciona al operador una mayor precisión y la retroalimentación durante el procedimiento. ProMIS ofrece una amplia gama de habilidades y procedimientos.

Figura 1.12: CAE ProMIS.
Fuente: Tomada de [40].

1.3.3. RoboSurgery

Basados en los modelos de Lapbot e Hibou en la Universidad del Cauca se han implementado versiones *software* para su simulación en un ambiente quirúrgico. Para el sistema robótico Lapbot se desarrolló un ambiente tridimensional en Ogre3D y Visual C++ [32]

En el año 2011 se realizaron dos proyectos que en conjunto desarrollaron un sistema virtual para el posicionamiento del robot Hibou. El sistema virtual fue hecho en Ogre3D y la posición del endoscopio es controlada por un casco que porta el cirujano. Dicho casco está formado por un sistema electrónico basado en un acelerómetro, el cual permite mover y orientar el órgano terminal al interior del paciente [41].

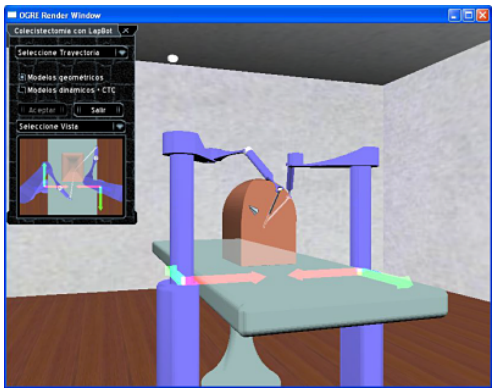

Figura 1.13: Ambiente 3D en Ogre3D con dos Lapbot.
Fuente: Tomada de [32].

En 2013 se desarrolló un *software* que integra la utilización de robots quirúrgicos en un entorno virtual 3D y que lleva por nombre RoboSurgery. La herramienta ha sido diseñada para que los ingenieros puedan comprender el uso de los asistentes robóticos en operaciones de laparoscopia. Integra dos tipos de robots: uno portaendoscopio (robot Hibou) y dos quirúrgicos (robots Lapbot), manipulados mediante un joystick.

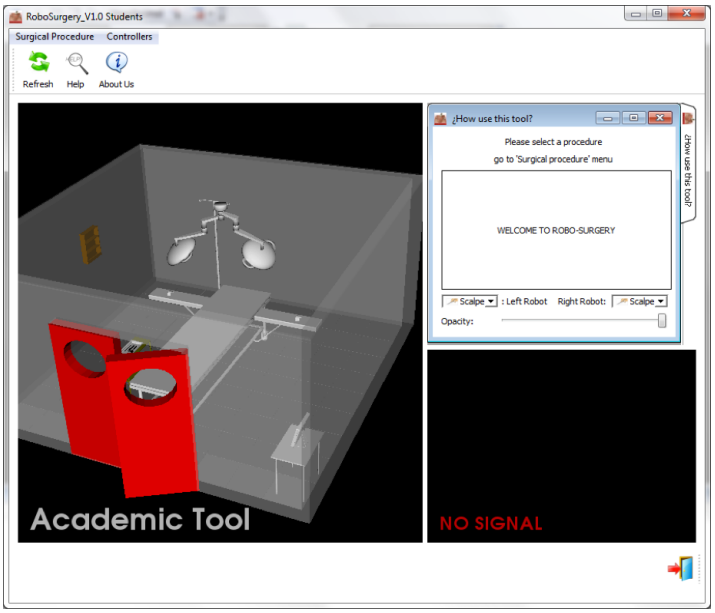

Figura 1.14: Ventana principal *Software* Robosurgery.
Fuente: Tomada de [42].

El sistema permite observar en una ventana el interior del abdomen del paciente. Una imagen virtual es generada por el endoscopio situado en el órgano terminal del robot porta-endoscopio Hibou, y mover el mismo internamente, esta forma de uso tiene como propósito realizar únicamente diagnóstico, por otro lado, el sistema permite la manipulación adicional de dos robots quirúrgicos con el fin de realizar una colecistectomía completa (extracción de la vesícula) [42].

Capítulo 2

Estructura mecánica de los robots y diseño de la tarjeta electrónica

Los robots porta endoscopio Hibou y Lapbot para cirugía laparoscópica (ver figura 2.1), son resultado de diversos trabajos de investigación, tesis de pregrado y maestría de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca. Estos robots fueron modelados de forma cinemática, dinámica, diseñados en CAD, simulados, y finalmente construidos [32, 43, 44].

En este capítulo se exponen los diferentes elementos tanto mecánicos como electrónicos, que han sido implementados en la construcción de los robots. Adicional se exponen brevemente las razones por las cuales los elementos que hacían parte de la tarjeta electrónica anterior han sido reemplazados presentando la nueva tarjeta electrónica con los elementos que harán parte de ella, el diseño y el prototipo terminado.

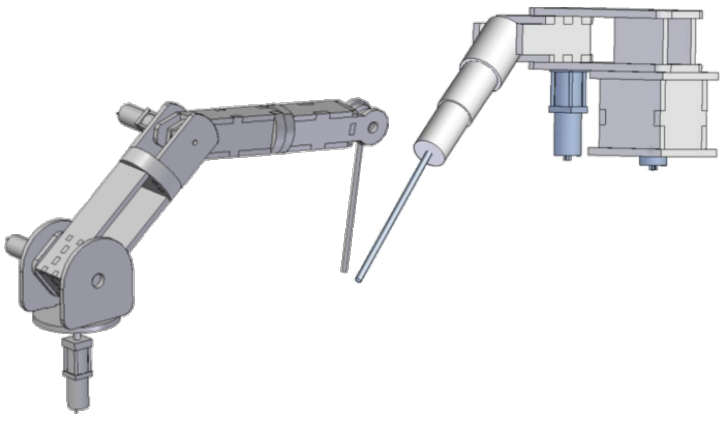

Figura 2.1: Hibou y Lapbot, Diseño CAD en SolidEdge®.
Fuente: Modificada de [43, 44].

2.1. Estructura mecánica de los robots

El Grupo de I+D en Automática Industrial desde hace algunos años ha desarrollado diferentes proyectos sobre robótica médica como por ejemplo el estudio de desempeño de un robot PUMA en laparoscopia, la simulación de un ambiente virtual de un robot SCARA, el modelado y simulación de un robot para laparoscopia, y el modelado y control de una mano robótica entre otros.

En la sección 1.2 se dio a conocer el desarrollo del robot porta endoscopio Hibou y el robot para cirugía laparoscópica Lapbot. Estos han sido los primeros pasos que encaminaron al desarrollo y construcción de un sistema de entrenamiento quirúrgico para cirugía laparoscopia en la Universidad del Cauca y que en su fase anterior concluyó con la construcción de los primeros prototipos reales.

2.1.1. Material y piezas de construcción

Los robots están construidos en un tipo de madera contrachapada llamada triplex, la cual es usada cotidianamente en carpintería y en construcción de interiores y exteriores, los motores de las articulaciones rotoides son de 12VDC, velocidad nominal de 75 RPM y un torque máximo de 15.6 N.m, su sensor asociado es un potenciómetro *Vishay* $533 - 11103$ con aproximadamente $0{,}4^o$de precisión [45]. El motor para articulación prismática es un *Linak* $LA314100$ de 300mm de desplazamiento [46], su sensor asociado es un potenciómetro lineal de $10K\Omega$ de impedancia. Los criterios de selección de estos elementos se explican en [43, 44].

2.1.2. Diseño y elementos anteriores

Hibou y Lapbot cuando fueron desarrolladas en sus respectivas tesis de grado, cada uno necesitaba tres tarjetas electrónicas para su funcionamiento (adquisición de datos y control, potencia y control de PWM).

En el diseño anterior de la tarjeta electrónica para adquisición y control de datos (ver figura 2.2), el dispositivo implementado para realizar las labores de control y adquisición de datos era un microcontrolador PIC18F4550[47] y el actuador un *"Pololu high-power motor driver 18V15"*[48]. La tarjeta de adquisición y control de datos tenía como criterios de funcionamiento contar con conexión mediante protocolo de comunicación USB, tener conversores análogos-digitales y capacidad de utilizar salidas digitales para manipulación de los motores. Por otro lado el actuador estaba diseñado para funcionar con voltajes desde 5.5VDC a 30VDC y entregar hasta 18VDC a 15A de forma continua.

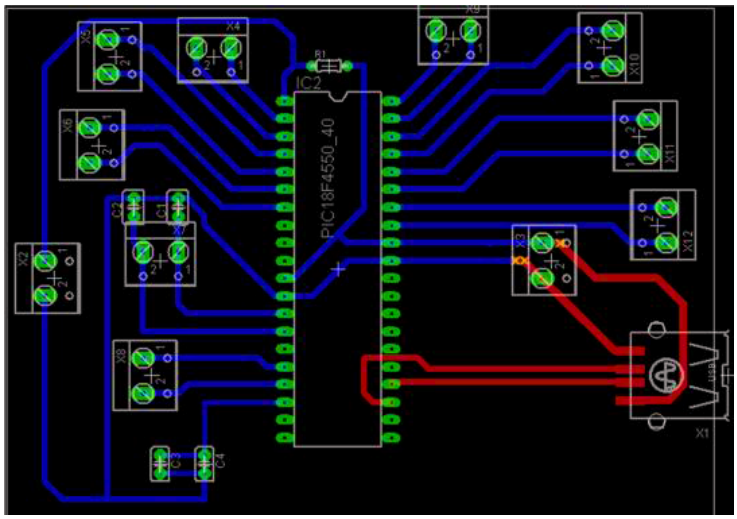

Figura 2.2: Diseño anterior de la tarjeta de adquisición de datos y control (No incluye potencia).

Fuente: Tomada de [43].

El microcontrolador PIC18F4550 cumplía ampliamente con sus exigencias, sin embargo requiere de dispositivos adicionales para su programación, periféricos para puesta en funcionamiento y comunicación por USB, además que usar todos estos elementos periféricos y no estar propiamente ensamblados son susceptibles al ruido electromagnético generando errores en el desempeño del mismo. De igual forma, teniendo como base la experiencia lograda trabajando con este actuador, no se tendrá en cuenta en este proyecto debido a que es muy susceptible a variaciones de tensión, lo cual ocasiona que su circuito integrado se queme con gran facilidad.

El uso de tres tarjetas electrónicas (adquisición de datos y control, potencia y control de PWM) se traduce en consumo energético, complejidad en el uso y espacio dentro del ambiente en donde se ubicaran los robots, además de no cumplir con criterios técnicos necesarios para este proyecto. Como primera medida los robots Hibou y Lapbot en sus respectivos trabajos de grado para sus construcciones contaban con un control de tipo *"On-Off"*, teniendo en cuenta que los robots deben realizar movimientos suaves se deben enviar señales PWM a los actuadores que permitan transmitir potencia a los motores de forma regulada, sin embargo el microcontrolador PIC18F4550 solo posee dos salidas de PWM y se necesitaba de tres señales para controlar los tres motores de cada robot, es decir seis salidas PWM en total. El diseño anterior que incluía el control *"On-Off"* transmitía una señal de PWM limitado hacia los actuadores, sin embargo estas señales PWM eran generadas mediante una tarjeta electrónica adicional que tenía un arreglo circuital con integrados LM555, donde la activación o no de esta señal era controlada mediante una

señal digital enviada desde el microcontrolador PIC18F4550. Esta señal de PWM tenía un ancho de pulso fijo, y la única forma de modificación era ajustando los elementos físicos del arreglo, razón por la cual no se podía utilizar para realizar control de salida variable.

2.2. Elementos de la nueva tarjeta electrónica

2.2.1. Actuadores

Para el desarrollo de este proyecto se llamará actuador a un puente-h, dispositivo electrónico capaz de invertir el sentido de giro a los motorreductores y de transferir potencia eléctrica de acuerdo a la cantidad de energía que el controlador desee suministrar.
El HMD-02A *"high power motor driver module"*(ver figura 2.3) es un puente-h capaz de soportar picos de corriente de hasta 90A, su placa está diseñada para funcionar con voltajes 5 VCD y trabajar con voltajes a la salida de 3 a 25 VCD, y hasta 35A continuos. Su ciclo de trabajo de PWM es aproximadamente $0 \sim 98\,\%$ de su voltaje de salida.

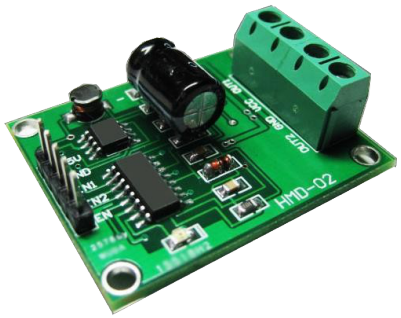

Figura 2.3: Puente H HMD-02A.
Fuente: Modificada de [49].

Estos puentes-h se utilizan debido a su gran desempeño y fidelidad durante el desarrollo de proyectos en laboratorios durante la carrera, además de que cumplen ampliamente con las necesidades de voltaje y corriente necesarios para el funcionamiento de los motorreductores y el motor de desplazamiento lineal.

2.2.2. Tarjeta de adquisición de datos y control

Actualmente en el mercado existen diversas plataformas embebidas que contienen todo lo considerado necesario para la implementación y funcionamiento de un controlador para los robots Hibou y Lapbot. De las múltiples opciones tecnológicas que se investigaron, se ha

elegido Arduino como el proveedor para el desarrollo de este proyecto, y como elemento principal el Arduino Mega 2560. El Arduino Mega 2560 (ver figura 2.4) es una placa *hardware* electrónica con un microcontrolador AtMega2560 que tiene 39 entradas o salidas digitales, 16 entradas analógicas con conversor análogo-digital de 10 bits de resolución y 13 salidas PWM de 8 bits de resolución. Este dispositivo puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa, además de permitir cargar un nuevo código sin el uso de un programador de *hardware* externo, en comparación del microcontrolador de los proyectos anteriormente mencionados.

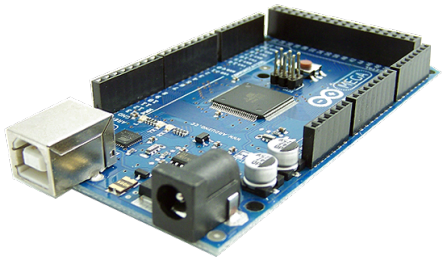

Figura 2.4: Arduino Mega 2560.
Fuente: Modificada de [50].

El Arduino Mega 2560 adicional cuenta con cuatro puertos seriales de comunicación, uno es el encargado de cargar los códigos de los programas, comunicar de forma serial y alimentar el dispositivo. Los puertos restantes solo permiten comunicación de datos.

2.2.3. Fuentes de Voltaje

Los elementos electrónicos implementados para realizar medición, control y transferir potencia a los motores tienen exigencias diferentes de voltaje para polarización y corrientes de funcionamiento (ver tabla 2.1).

Elementos	**Voltaje (V)**	**Corriente max (mA)**
Sensores Vishay	12	240
Puente-h HMD-02	5	100
Motor lineal	24	1300
Motor rotatorio	12	5500
Arduino Mega 2560	5∼12	500

Tabla 2.1: Voltajes de polarización.

Para alimentar los robots y la tarjeta integrada se cuenta con tres fuentes. Dos de las fuentes son de referencia ATX 750W de UNITEC. Estas fuentes han sido diseñadas para

uso en computadores de escritorio, sin embargo se ha adaptado para ser utilizada en este proyecto. Los datos de la placa de fábrica se pueden apreciar en la tabla 2.2.

	Voltaje (V)	**Corriente (A)**
Power Input AC	115	8
Power Output DC	3,3	31
	5	31
	12	40
	-5	0.5
	-12	0.5

Tabla 2.2: Datos de la placa de fábrica, fuente de voltaje ATX 750W.

De acuerdo a la información que ofrece la placa de la fuente de alimentación, estas fuentes tiene la capacidad de suministrar el voltaje y la corriente necesaria para los elementos electrónicos y los motores de los robots, salvo el motor lineal. La tercera fuente, es un cargador de *"siemens"* diseñado para uso en computadores portátiles, sin embargo se ha adaptado para ser utilizada en este proyecto siendo capaz de entregar 19VDC y hasta 4A. Esta fuente se encarga de alimentar la salida del actuador que esta conectado al motor lineal. La distribución de las fuentes de voltaje se realiza de tal forma que los elementos de medición y control han sido separados de los de potencia debido a que al exigir corrientes elevadas por parte de los motores, el voltaje de polarización de los sensores caía realizando fluctuaciones en la medición que imposibilitaban un correcto control. Por esta razón una de las fuentes ATX es utilizada para alimentar los sensores y acoplar tierras con el Arduino Mega 2560, la otra fuente ATX para alimentar los actuadores y los motores rotacionales y finalmente el cargador que alimenta el motor lineal.

2.3. Diseño e implementación de nueva tarjeta electrónica

Como objetivo de este trabajo de grado se tiene implementar un nuevo *hardware* de adquisición de datos, control para los robots Hibou y Lapbot, y comunicación con el computador. Las tarjetas electrónicas anteriores requerían demasiadas conexiones y sus diseños hacían que éstas no fueran de ayuda en este proyecto. En el nuevo diseño que integra la adquisición de señales de los sensores, el control de las articulaciones de los robots y la comunicación con la interfaz gráfica. La tarjeta electrónica sirve como centro de distribución, hacia esta van todas las señales de voltaje de las fuentes, las señales de los sensores y el cableado de los motores. La tarjeta se encarga de acoplar las señales de tierra (GND) de la fuente de voltaje para los sensores con el Arduino Mega 2560, así como de regular el voltaje para los actuadores. En la figura 2.5 se muestra el esquema de conexión de una de las articulaciones (motor y sensor) en la tarjeta electrónica, las otras cinco articulaciones poseen el mismo esquema de conexión con diferentes pines al Arduino Mega 2560.

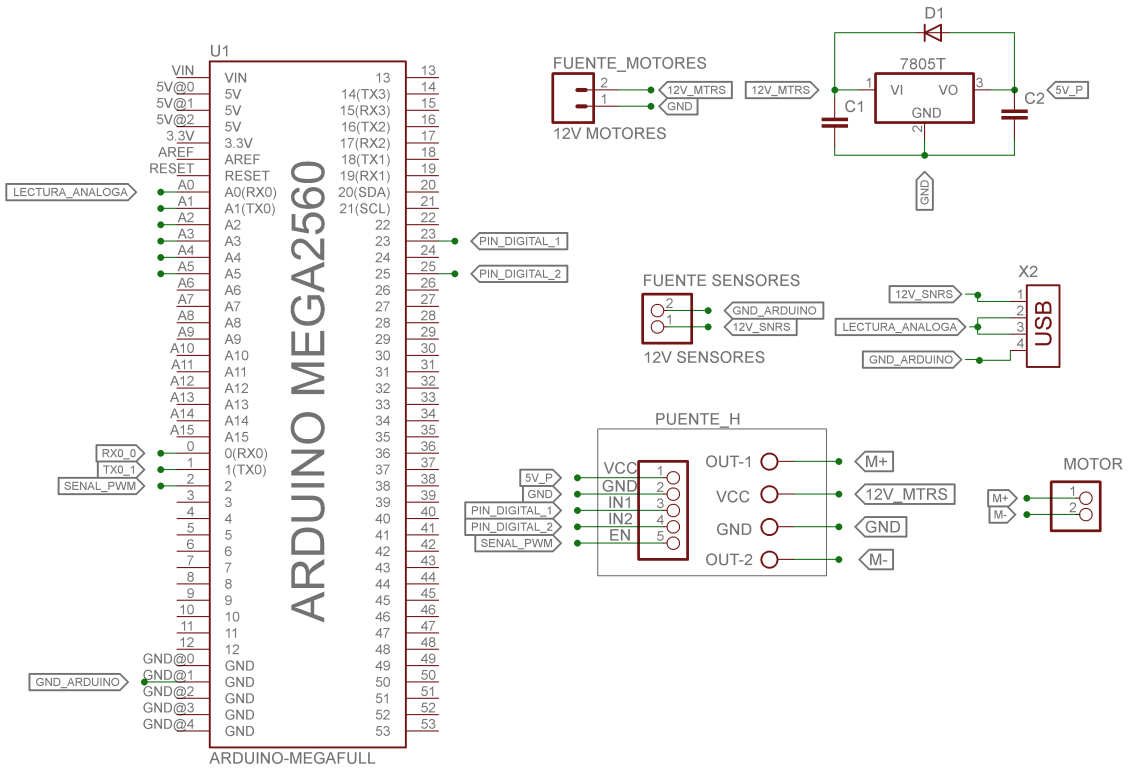

Figura 2.5: Circuito esquemático de una articulación en la nueva tarjeta electrónica. Fuente: Elaboración propia.

El diseño de la tarjeta electrónica se ha realizado en *" CadSoft EAGLE PCB"* (ver figura 2.6), de tal forma que el cableado sea de tipo *"Plug and Play"*, esto significa que el Arduino Mega 2560 puede ser removido o reemplazado por el usuario, los sensores de las articulaciones son de conexión USB, los actuadores están conectados por *"jumpers"* y los motores se conectan por borneras.

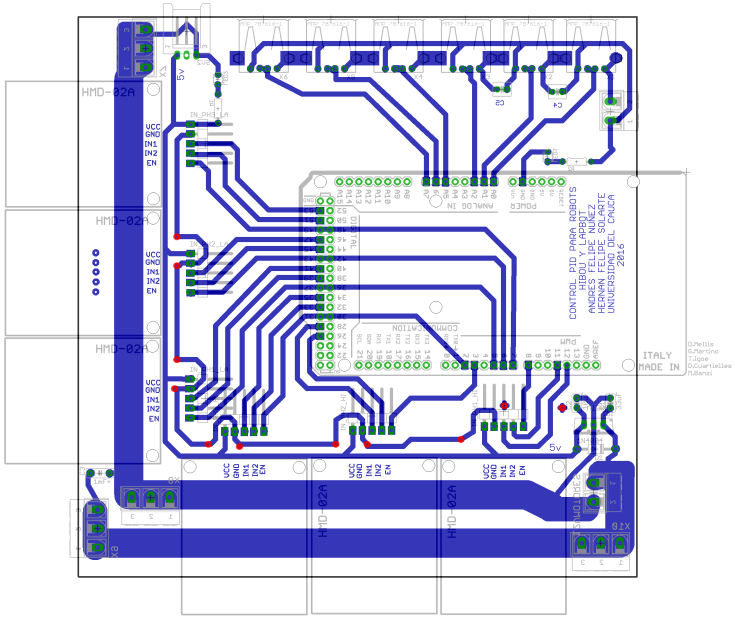

Figura 2.6: Diseño en *"CadSoft EAGLE PCB"* de la tarjeta electrónica.
Fuente: Elaboración propia.

	Sensores	Puentes-h		
	ADC	EN	IN2	IN1
HIBOU ART 1	A0	8	11	12
HIBOU ART 2	A1	3	27	29
HIBOU ART 3	A2	2	33	35
LAPBOT ART 1	A5	5	39	41
LAPBOT ART 2	A6	6	45	47
LAPBOT ART 3	A7	7	51	53

Tabla 2.3: Pines de conexión del Arduino Mega 2560 a los periféricos.

Los pines de conexión del Arduino Mega 2560 (ver tabla 2.3) a los sensores y los actuadores se han caracterizado para ser considerados en el controlador. La Arduino Mega 2560 se ha fijado a la tarjeta electrónica, los puertos USB de los sensores han sido conectados, los actuadores conectados mediante *"jumpers"* y finalmente los motores conectados a borneas las cuales están conectadas a las salidas de los actuadores, teniendo como resultado una tarjeta electrónica integrada, completamente ensamblada y funcionando (ver figura 2.7).

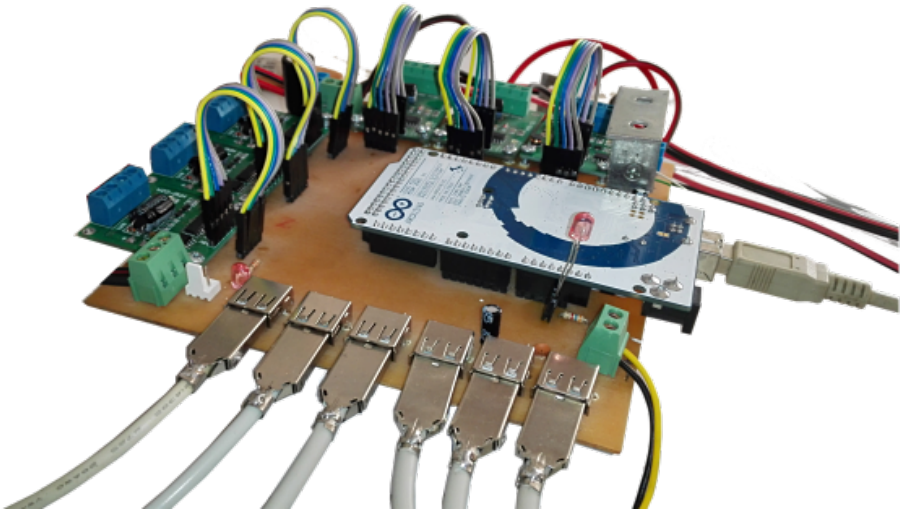

Figura 2.7: Nuevo diseño tarjeta electrónica.
Fuente: Elaboración propia.

Capítulo 3

Controlador PID

Para desarrollar el controlador PID para los robots Hibou y Lapbot, inicialmente se simula en Matlab los modelos de los robots y se sintonizan de forma manual las constantes de los controladores para cada articulación, verificando sus respuestas para diferentes consignas.

En este capítulo se detalla de forma general los aspectos necesarios para realizar la implementación del controlador PID en el Arduino, teniendo como punto de partida las simulaciones previamente realizadas. Se caracterizan los sensores y se explica la forma en las que se hace medición, de igual forma se explica el tiempo de muestreo utilizado para la ley de control y la forma en que se desarrolla e implementa el derivador del controlador, finalizando con la configuración necesaria para las señales de PWM aplicadas a los actuadores con el fin de mejorar la movilidad de los robots.

3.1. Esquema general del controlador PID

El controlador PID detallado en esta sección es ampliamente utilizado en robótica industrial para robots tipo serie. El controlador PID tipo paralelo es considerado como un sistema lineal y cada articulación es controlada por un control descentralizado de tipo PID con ganancias constantes. Su ventaja es la facilidad de implementación y su bajo costo. En contraste se pueden encontrar malas precisiones y desplazamientos excesivos en el caso de movimientos rápidos. Su esquema general se aprecia en la figura 3.1.

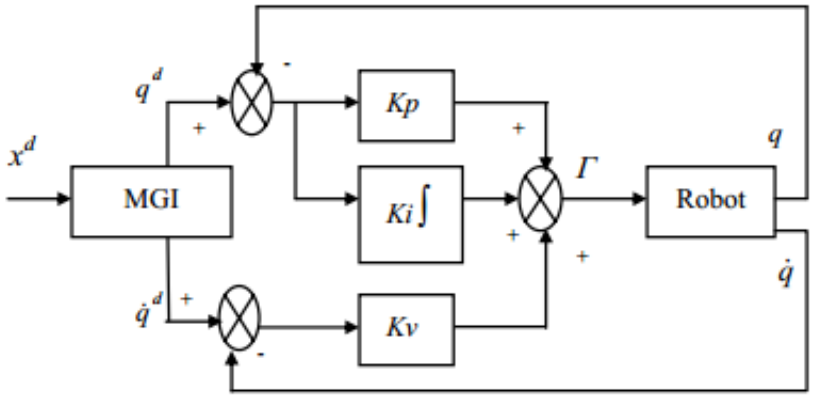

Figura 3.1: Estructura del controlador PID.
Fuente: Tomada de [51].

En este estudio se realiza el control individual por cada articulación, es decir que la señal de control es afectada y afecta a una sola articulación. La relación de salida de cada articulación es descrita por la ecuación 3.1:

$$\Gamma_i = Kp_i(q_{des} - q_{real}) + Kv_i(\dot{q_{des}} - \dot{q_{real}}) + Ki_i \int (q_{des} - q_{real})d\tau \tag{3.1}$$

Para la ecuación 3.1 es necesario obtener la velocidad de cada una de las articulaciones. Un modelo dinámico directo[1] típico en simulación permite obtener las posiciones y velocidades de salida, mientras que las velocidades deseadas se obtienen derivando las posiciones deseadas con un bloque de derivación $dq/d\tau$ mediante Simulink, sin embargo en implementaciones no se tiene una medida de las velocidades reales, por tal razón se realiza la implementación del bloque de derivación descrito en la ecuación 3.2 y que tiene por respuesta en frecuencia el diagrama de la figura 3.2.

$$G_d(s) = \frac{T_d s}{\alpha T_d s + 1} \tag{3.2}$$

Donde $\alpha = 0{,}01$ y $T_d = 1$.

[1]Modelo dinamico directo: $\ddot{q} = f(q, \dot{q}, \Gamma, f_e)$. Donde: Γ = fuerza aplicada a los actuadores, f_e = esfuerzo exterior que ejerce el robot sobre el ambiente, $q\ \dot{q}\ \ddot{q}$ = posiciones, velocidades y aceleraciones respectivamente.

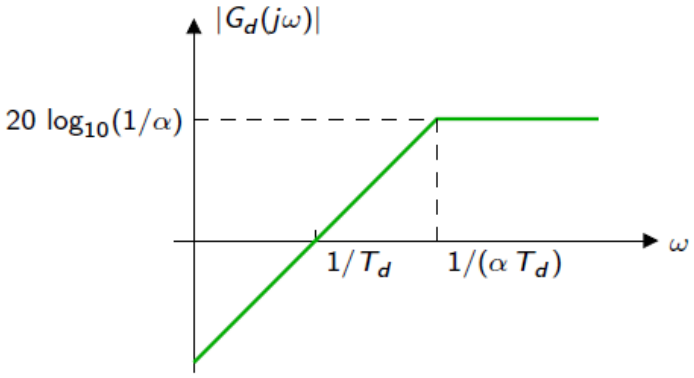

Figura 3.2: Ganancia del derivador con un polo.
Fuente: Tomada de [52].

El modo derivativo está representado por un derivador real, que incluye un filtro paso bajo en el que la constante α permite seleccionar la posición del polo. Este tipo de modelo permite eliminar la amplificación de ruido que se genera con un modelo de derivador puro.

Como regla general el polo se encuentra ubicado aproximadamente diez veces a la izquierda del cero, por lo tanto su influencia en la respuesta del controlador es despreciable, razón por la cual no es necesario incluirla en los estudios analíticos de los lazos de control.

3.2. Simulación en MATLAB - Simulink

La simulación de los robots con el controlador PID se realiza con el esquema control PID cartesiano detallado en [51] el cual se aprecia en la figura 3.3. El código del modelo dinámico así como los modelos geométricos directo e inverso de los robots Hibou y Lapbot son los implementados respectivamente en [32, 35]. Los valores de los parámetros inerciales de los modelos dinámicos de los robots son los calculados para Hibou en [43], y para Lapbot en [44].

El tiempo de simulación utilizado es $Ts = 0{,}01$ segundos. La simulación puede realizarse en un intervalo más pequeño, pero a nivel de implementación del tiempo de ejecución en la tarjeta electrónica, la aplicación gráfica y la comunicación entre ellos no permiten que el tiempo de muestro sea menor.

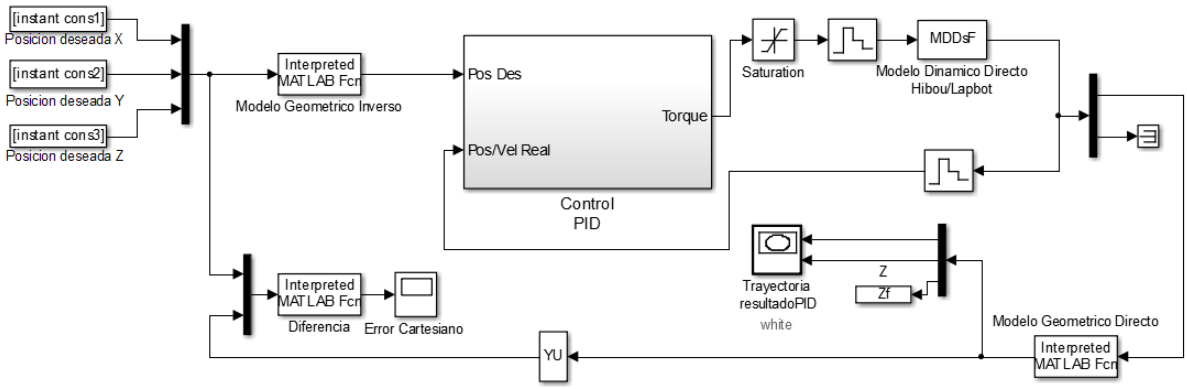

Figura 3.3: Esquema PID Cartesiano.
Fuente: Modificada de [51].

3.2.1. Método de sintonización

La sintonización de las variables para el controlador se realizan manualmente variando los parámetros Kp, Kv y Ki para cada articulación, siguiendo las recomendaciones planteadas en [51] hasta obtener un error articular pequeño.

Los valores que mostraron un mejor desempeño y cuyo resultado se muestra en la sección 3.2.3 son:

Hibou

- Constante proporcional Kp: $[700 - 800 - 100]$.
- Constante de velocidad Kv: $[20 - 15 - 6]$.
- Constante integral Ki: $[0 - 10 - 100]$.

Lapbot

- Constante proporcional Kp: $[1500 - 300 - 220]$.
- Constante de Velocidad Kv: $[100 - 11 - 13]$.
- Constante integral Ki: $[200 - 0 - 150]$.

3.2.2. Espacio de trabajo

Hibou

Los parámetros máximos y mínimos del espacio de trabajo del robot Hibou son los presentados a continuación. Las distancias se toman teniendo como referencia el eje de la primera articulación.

- Posición del trocar: $Tx = 40$ cm, $Ty = 0$ cm, $Tz = 23$ cm.
- Amplitud de desplazamiento en eje x: 7cm desde el origen del trocar (Min: 33cm y Max: 47 cm).
- Amplitud de desplazamiento en eje y: 7cm desde el origen del trocar (Min: -7 cm y Max: 7 cm).
- Amplitud de desplazamiento en eje z: 15cm desde el origen del trocar (Min: 23 cm y Max: 8 cm).

Lapbot

El órgano terminal introducido en la cavidad abdominal puede alcanzar una distancia máxima de 20 cm desde el punto de incisión, su posición de inicio está a 10 cm a partir de este punto de incisión.

- Posición del trocar: $Tx = 30$ cm, $Ty = 13{,}7$ cm, $Tz = 20$ cm.
- Amplitud de desplazamiento en eje x: Min: 20 cm y Max: 40 cm.
- Amplitud de desplazamiento en eje y: Min: 13,7 cm y Max: $-6{,}3$ cm.
- Amplitud de desplazamiento en eje z: Min: 0 cm y Max: 20 cm.

3.2.3. Resultados de simulación

Consigna tridimensional - Forma helicoidal

La consigna tridimensional se define como una línea curva y continua que gira uniformemente en torno al eje Z realizando la forma de un cilindro, esta avanza en las tres dimensiones: las distancias de los ejes X y Y al centro de la hélice son siempre las mismas, definidas por un radio R y varía la altura en el eje Z respecto a su punto de inicio.

Los parámetros de esta consigna se han definido de acuerdo al espacio de trabajo de los robots Hibou y Lapbot (ver tabla 3.1).

	Hibou	**Lapbot**
Radio [cm]	4	4
Centro x [cm]	40	30
Centro y [cm]	0	4
Mov z [cm]	11-15	0-4
Tiempo simulación [s]	10	10
Tiempo de muestreo [s]	0.01	0.01

Tabla 3.1: Parámetros de la consigna helicoidal para simulación en Simulink.

Movimiento en la articulación final

La figura 3.4 muestra la trayectoria realizada por la articulación final, según los parámetros especificados anteriormente.

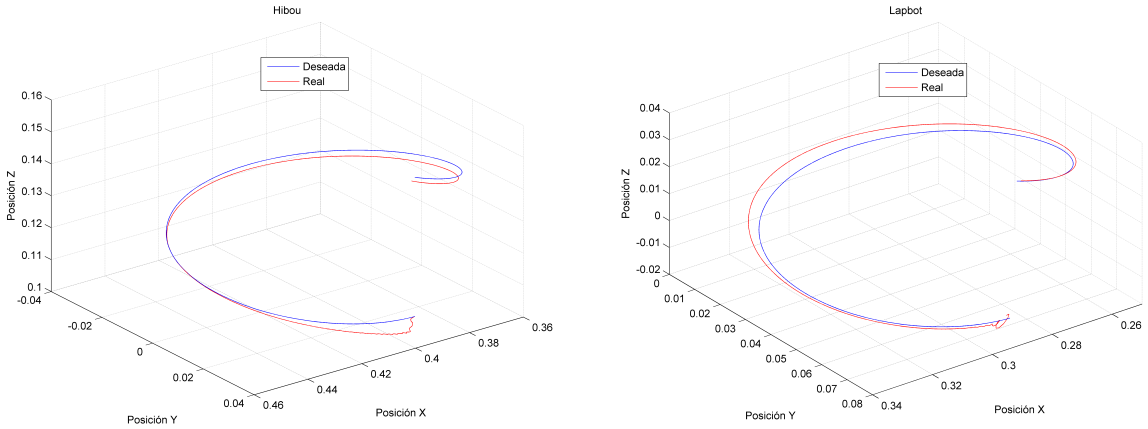

Figura 3.4: Trayectoria helicoidal realizada por la articulación final.
Fuente: Elaboración propia.

Error cartesiano

Para los robots Hibou y Lapbot se observa un error cartesiano inferior a 2 mm y 8 mm respectivamente (ver figura 3.5).

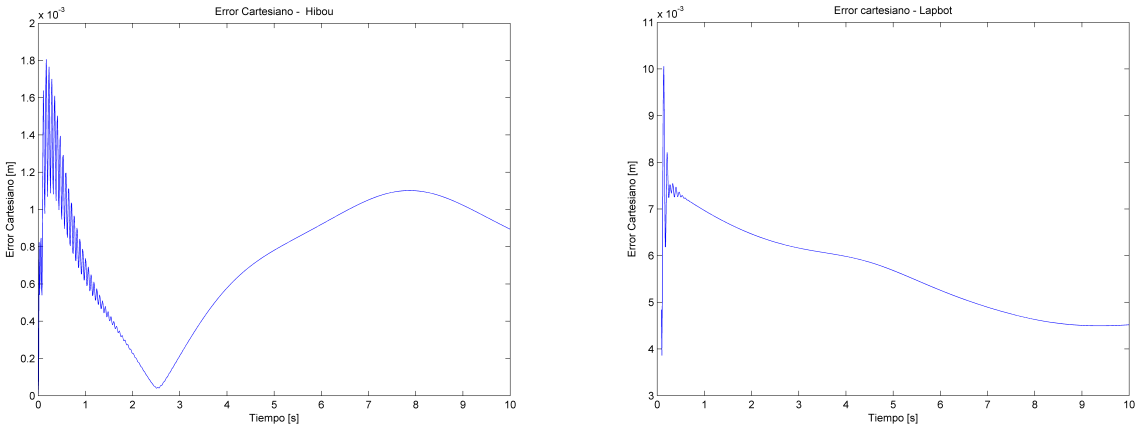

Figura 3.5: Error cartesiano trayectoria helicoidal.
Fuente: Elaboración propia.

3.3. Implementación en Arduino

3.3.1. Puerto serial

La implementación sobre la tarjeta Arduino Mega 2560 hace necesaria la transmisión y recepción de datos usando el puerto serial. Debido a la actualización y lectura de datos, realización de la rutina de control y exigencia del cumplimiento del tiempo de muestreo, se hace necesario que la velocidad de transmisión sea tan alta como sea posible de tal forma que no se generen retardos al recibir o enviar información. Por dicha razón se ha escogido configurar la velocidad de transmisión en 115200 baudios, la cual es la velocidad máxima permitida para utilizar sobre este dispositivo. Los demás parámetros utilizados para configurar un puerto serial se dejan con valores por defecto: número de datos de envió de 8 bits, sin bit de paridad y un bit de *stop*[50].

Al usar la máxima velocidad soportada por el *hardware* Arduino se debe garantizar una distancia corta entre los dos dispositivos a comunicar, de lo contrario se pueden generar inestabilidades y perdidas en la transmisión de datos [53].

Interrupción serial - *Serial Event*

La imposibilidad de sincronizar los procesos del Arduino y el computador debido a diferentes ciclos de tiempo para la ejecución de tareas y carga computacional, originan que el envío y recepción de información entre los dispositivos no se pueda realizar de forma tradicional.

Las interrupciones son la forma más común de conocer e informar eventos externos ya sea tipo *hardware* con pines asociados que detectan el cambio de una señal digital, o por efecto de un *timer* como la finalización de un ciclo de trabajo. Estas interrupciones sobre eventos externos permiten realizar una acción en respuesta a este evento, la velocidad de estas acciones varía de acuerdo a su prioridad y se hacen en espacios de código independiente al que se esté ejecutando en ese momento en el micro controlador[54].

El Arduino posee un evento llamado *Serial Event* que se activa una vez se detecten datos de entrada por el pin RX del puerto serial. *Serial Event* ejecuta un segmento de código de modo casi inmediato, permitiendo que la entrada de datos sea leída y procesada evitando errores y perdida de datos. Las interrupciones de todo tipo en Arduino permiten una vez ejecutado el segmento de código del evento regresar al sitio donde se encontraba previo a la interrupción.

3.3.2. Tiempo de muestreo

Las características de un sistema de control en tiempo discreto dependen del periodo de muestreo T_s. Un periodo de muestreo grande tiene efectos dañinos sobre la estabilidad relativa del sistema, por ello hay que tener especial atención a la hora de elegir un valor adecuado. Además se debe garantizar un tiempo de muestro constante para obtener un buen desempeño de la aplicación.

El tiempo transcurrido entre instantes de muestra, control y envió de datos se toma en base al tiempo interno de la tarjeta Arduino proporcionado por la función *millis()*; esta hace uso del contador interno del microcontrolador ATmega. Este contador se incrementa en cada ciclo de reloj, el cual es controlado por un cristal.

La precisión del cristal puede variar dependiendo de la temperatura y la tolerancia propia del cristal. Se ha reportado una variación entre 3 y 4 segundos por un periodo de 24 horas, lo cual es suficiente para nuestra aplicación y confirma que la tarjeta Arduino y su contador interno permite garantizar el tiempo de muestreo estable e igual a 10 ms para el controlador [55].

3.3.3. Arranque modificado para motores DC

Los movimientos de las articulaciones de los robots deben ser suaves y a baja velocidad en todo momento. Los motorreductores con los que se cuenta poseen una caja reductora con una relación de reducción tan pequeña que no permite por si sola realizar movimientos suaves y precisos, además de que al ser motores de alto torque de salida en su eje exigen corrientes muy altas para su arranque.

Bajo pruebas de ensayo y error se ha encontrado el valor nominal de PWM capaz de arrancar los motorreductores, pero este es demasiado grande y hace que los motores inicien con velocidades altas impidiendo estabilizar el robot cuando el error es muy pequeño. De igual forma intentar un valor de PWM más pequeño a este, hace que los motores sean incapaces de arrancar al torque mínimo necesario para romper fricción.

Con el fin de solventar este inconveniente, se realizan dos modificaciones a la señal PWM, las cuales se describen a continuación.

Modificación de frecuencia

En [56] se plantea una solución a este problema. Una reducción en la frecuencia del PWM genera un aumento en la corriente de arranque con voltajes bajos, lo que permite aumentar el torque necesario para darle movimiento al motor y al mismo tiempo disminuir su velocidad de arranque.

La frecuencia de la señal PWM generada por el Arduino Mega 2560 depende de varios *"timers"* que adicionalmente controlan los ciclos de trabajo, la activación de las ondas y el tipo de la señal. El Arduino Mega 2560 tiene 15 salidas PWM: pines 2 al 13 y 44 al 46 y cuenta con un total de cinco *"timers"*, los cuales tienen asociado una cantidad de estos pines PWM (ver tabla 3.2).

	Pines		
Timer0	4	13	
Timer1	11	12	
Timer2	9	10	
Timer3	2	3	5
Timer4	6	7	8
Timer5	44	45	46

Tabla 3.2: Pines asociados a los *"timers"* para PWM en Arduino Mega 2560.

Las salidas PWM para cada *"timer"* tienen normalmente la misma frecuencia, pero pueden tener diferentes ciclos de trabajo. La frecuencia para todos los *"timers"* por defecto es 490 Hz, excepto para el *"timer0"* cuya frecuencia es 980 Hz.

Para acceder a los registros del *"timer"* y realizar la correspondiente variación de la frecuencia desde código se modifica el registro TCCRnB de 8 bits, donde *"n"* es el número del *"timer"*.

En TCCRnB los tres primeros bits de derecha a izquierda se llaman CSn2, CSn1 y CSn0, los cuales controlan la frecuencia de PWM del *"timer n"*, estos bits representan un número entero entre 0 y 7 llamado *" preescaler"*. Para realizar cambios en la frecuencia, los valores de *"preescaler"* y su frecuencia de resultado pueden ser consultados en la tabla 3.3.

La función *void setPWMPreescaler(int timer, int preescaler)* permite cambiar el valor de la frecuencia del PWM segun el *"timer"* seleccionado y un valor predefinido para *prescale* según la tabla 3.3.

Timer0		
Valor	**Preescaler**	**Frecuencia (Hz)**
0x01	1	62720
0x02	8	7840
0x03	64	980
0x04	256	245
0x05	1024	61,25

Timer 1,3,4,5		
Valor	**Preescaler**	**Frecuencia (Hz)**
0x01	1	31360
0x02	8	3920
0x03	64	490
0x04	256	122,5
0x05	1024	30,625

Timer2		
Valor	**Preescaler**	**Frecuencia (Hz)**
0x01	1	31360
0x02	8	3920
0x03	32	980
0x04	64	490
0x05	128	245
0x06	256	122,5
0x07	1024	30,625

Tabla 3.3: Frecuencias para cada *"timer"* en Arduino Mega 2560.

Dependiendo del valor *preescaler* elegido, su equivalente en binario se cargará en la variable *mode*. Inicialmente los tres bits del registro TCCRnB son restablecidos a 000 usando la operación *AND*, y finalmente se añaden los valores cargados en *mode* al registro TCCRnB usando la operación *OR*.

```
void setPWMPreescaler(int timer, int preescaler) {
 if (timer == 1){
   switch(preescaler){
           1: mode = 0b001;
           8: mode = 0b010;
          64: mode = 0b011;
         256: mode = 0b100;
        1024: mode = 0b101;
      }
  TCCR1B = (TCCR1B & 0b11111000) | mode;
  }
  if (timer == 2)
  ...
```

Se debe tener en cuenta que el *"timer0"* es el que controla las funciones de tiempo en Arduino, es decir si se cambia este *"timer"* las funciones como delay() o millis () seguirán trabajando pero a una escala de tiempo diferente, es decir más rápido o más lento, y ya no representarán un medida de tiempo real.

Modos PWM

La señal PWM tiene dos modos de generación de ondas: *"Fast PWM"* y *"Phase correct PWM"* (ver figura 3.6).

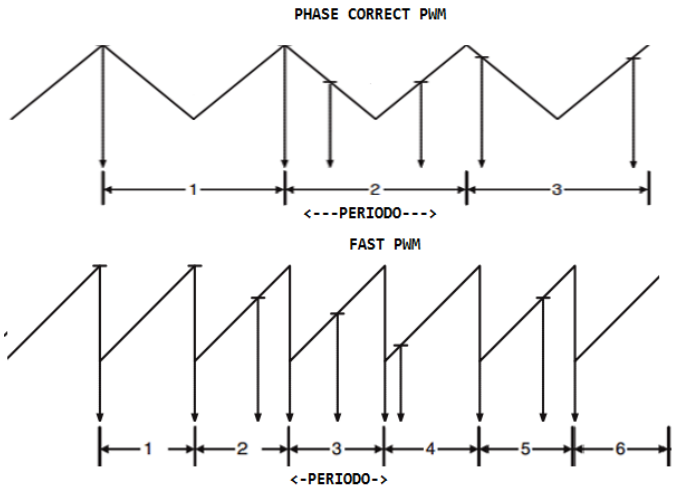

Figura 3.6: Modos de generación de onda PWM en Arduino Mega 2560.
Fuente: Modificada de [50].

- *"Fast PWM"*: Permite generar ondas PWM de alta frecuencia, este modo tiene en su operación una sola pendiente en la onda. La frecuencia de funcionamiento de este modo es el doble de alta que en el modo *"Phase correct PWM"*. La alta frecuencia hace que sea útil para la regulación de potencia, rectificación y aplicaciones de DAC.

- *"Phase correct PWM"*: Permite generar ondas PWM de alta resolución, este modo se basa en una operación de doble pendiente con flanco de subida y de bajada, la cual tiene una frecuencia máxima de operación menor que *"Fast PWM"*. Sin embargo debido a su característica simétrica de doble pendiente, resulta muy útil en aplicaciones de control de motores, razón por la cual se realiza la configuración respectiva para utilizar este modo en el Arduino Mega 2560.

De la misma manera como se modificaron los registros del *"timer"* para la frecuencia, para modificar el modo PWM se accede a el cuarto bit de derecha a izquierda del registro TCCRnB llamado WGMn2, y a los dos primeros bits de derecha a izquierda del registro TCCRnA llamados WGMn1 y WGMn0, donde *"n"* representa al *"timer"* asociado. Estos bits controlan los modos del PWM del *"timer n"*, los cuales se pueden configurar de acuerdo a la tabla 3.4.

Modo	**WGMn2**	**WGMn1**	**WGMn0**	**Modo de operación**
0	0	0	0	*Normal*
1	0	0	1	*Phase Correct*
2	0	1	0	*Phase Correct*
3	0	1	1	*Phase Correct*
5	1	0	1	*Fast PWM*
6	1	1	0	*Fast PWM*
7	1	1	1	*Fast PWM*

Tabla 3.4: Modos de onda de generación de PWM en Arduino Mega 2560.

La función *void setPWMWave(int timer, int wave)* permite cambiar el modo de operación según el *"timer"* seleccionado y el valor de *wave* que define el modo según la tabla 3.4.

Los valores y su equivalente binario se cargan en los registros TCCR1A y TCCR1B de la misma forma que los registros del *preescaler* detallados anteriormente. En el siguiente fragmento de código se realiza la modificación del modo de operación para el *"timer"* 1.

```
void setPWMWave(int timer, int wave){
  switch(wave){
        1: modoA = 0b01; modoB = 0b00000;
        3: modoA = 0b11; modoB = 0b00000;
        5: modoA = 0b01; modoB = 0b01000;
        7: modoA = 0b11; modoB = 0b11000;
     }
    if (timer==1) {
    TCCR1B = TCCR1B & 0b11100111 | modoB;
    TCCR1A = TCCR1A & 0b11111100 | modoA;
    ...
    }
  }
```

3.3.4. Valores de referencia

Todos los valores enviados se encuentran en unidades de grados [^{o}] y estan multiplicados por 10 con el fin de aumentar su resolución de envío. Los valores de la articulación 1 para el robot Lapbot se encuentran en centímetros [cm] y los valores de la articulación 1 para el robot Hibou se encuentran desfasados positivamente en 30º. Internamente en el programa Arduino o en la aplicación C++ se realiza la conversión a radianes para el controlador o el modelo geométrico inverso (MGI), dado que estos utilizan este tipo de unidades.

Todos los datos se envían en una misma cadena de datos, diferenciándose por una serie de banderas definidas de la siguiente forma:

- * : Articulación 1
- # : Articulación 2
- % : Articulación 3

Una trama completa de datos para Hibou es por ejemplo: * 300 # 833 % 233 ! donde el valor para la primera articulación seria igual a = 300/10 = 30º (corrección de resolución), luego $\rightarrow$ 30 − 30 = 0º(corrección de desfase aplicado), por tanto la articulación 1 = 0º, así con estos operaciones igualmente para la articulación 2 = 83,3º y la articulación 3 = 23,3º. El carácter ! se usa como verificador de fin de trama, permitiendo detectar errores en el envío.

3.3.5. Lectura de sensores

La lectura análoga de los sensores se realiza con la resolución estándar del Arduino Mega 2560, la cual es de 10 bits, esto indica que el valor entregado en las variables de lectura comprende un rango entre $0 - 1023$.

Como se describió en la sección 2.1 se utilizan dos tipos de sensores; con el fin de conocer su comportamiento se realiza la caracterización y gráfica de sus respuestas.

Potenciómetro rotativo

Como el sensor es lineal [43][44], se puede verificar su comportamiento de forma algebraica realizando operaciones matemáticas de acuerdo a las características y valores de funcionamiento.

- **Voltaje de Referencia** ($V_{ref} = 5V$): Voltaje de referencia que utiliza el conversor ADC del Arduino para la lectura de la entrada analógica.
- **Grados del potenciómetro** ($L_{pot} = 1080$): Máximo número de grados que pueden ser medidos con el potenciómetro rotativo.
- **Voltaje de alimentación del potenciómetro** ($V_{alm} = 12V$).
- **Resolución de ADC** ($ADC_{10BIT} = 1023$).

El máximo voltaje de entrada que soporta el ADC es $5V$, por lo tanto el máximo número de grados disponibles para la lectura es:

$$\frac{V_{ref}}{V_{alm}} L_{pot} = \frac{5}{12} \cdot 1080 \ [\frac{V}{V} \cdot^{\circ}] = 450^{\circ}$$

La medición es máximo de 450° respecto a los 1080° de rango total del sensor. Sin embargo estos son suficientes para el rango de movilidad de las articulaciones de los robots.

La resolución del potenciómetro en grados por voltio $[\frac{^{\circ}}{V}]$, la cual está dada por:

$$\frac{L_{pot}}{V_{alm}} = \frac{1080}{12} = \frac{90}{1} [\frac{^{\circ}}{V}]$$

El aumento de 90 grados en el potenciómetro presenta una variación de $1V$. Un grado equivale a una variación de 0,011 V .

La variación en voltios por cada *Cuenta*[2] tiene como resultado la resolución de lectura en el ADC del Arduino, la cual está dada por:

$$\frac{V_{ref}}{ADC_{10BIT}} = \frac{5}{1023}[\frac{V}{Cuentas}]$$

Por tanto para esta resolución, se tiene una variación de $0,0048V/cuenta$.

Encontrada la resolución de lectura del Arduino y del potenciómetro, se puede encontrar el número equivalente de grados de la lectura del ADC.

$$\begin{aligned} Grados &= \frac{\text{Resolución del potenciómetro}}{\text{Resolución de lectura}} \\ &= \frac{90}{1} \cdot \frac{5}{1023}[\frac{^\circ}{V}\frac{V}{Cuentas}] = \frac{150}{341}[\frac{^\circ}{Cuentas}] \end{aligned}$$

Este valor es la pendiente que describe la ecuación de una línea recta entre la posición angular y las cuentas del ADC del arduino. La ecuación final con la conversión de grados a radianes, se hace de acuerdo a la ecuación 3.3.

$$Radianes = \frac{150}{341} \cdot ADC \cdot \frac{\pi}{180} \tag{3.3}$$

Como método de verificación, estos datos se procesaron mediante el *software* MATLAB usando la función *polyfit*[3] para un ajuste de un polinomio de grado 1 o ecuación de línea recta (ver figura 3.7).

[2]Cuenta = Resolución entregada por el conversor análogo digital, indica el menor paso de cuantificación para la señal de entrada. Un ADC de 10bits tiene 1023 cuentas.

[3]p = polyfit(x,y,n). Devuelve los coeficientes de un polinomio p(x) de grado n que realicen un mejor ajuste para los datos en y (en un sentido de mínimos cuadrados). Los coeficientes de retorno están en potencias descendentes, y la longitud es n + 1.

$$p(x) = p_1x^n + p_2x^{n-1} + ... + p_nx + p_{n+1}$$

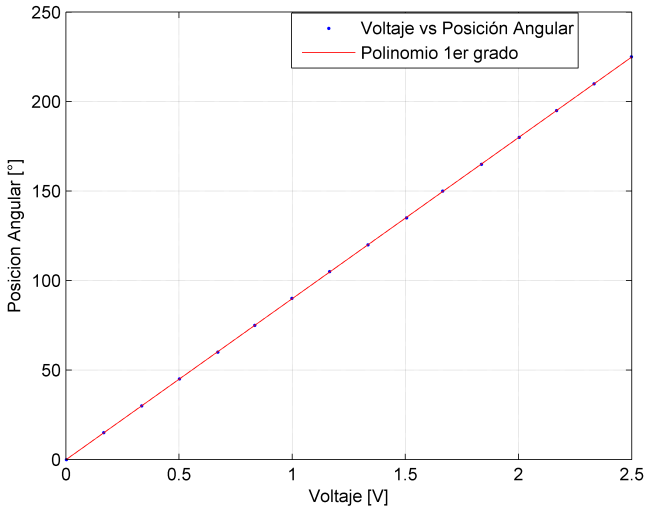

Figura 3.7: Curva voltaje vs posición angular potenciómetro rotativo.
Fuente: Elaboración propia.

Potenciómetro lineal

El potenciómetro lineal está alimentado a 5VDC debido a que se utiliza todo su rango de medición, contrario a los potenciómetros rotativos. Se realiza una toma de datos experimental de voltaje y distancia (ver tabla 3.5) para conocer su comportamiento.

Distancia (mm)	Voltaje (V)	Distancia (mm)	Voltaje (V)	Distancia (mm)	Voltaje (V)
60,00	5,00	43,57	3,18	18,36	1,34
59,13	4,85	41,58	3,01	15,87	1,17
57,88	4,68	39,52	2,84	13,51	1,00
56,51	4,52	37,41	2,68	11,14	0,84
55,08	4,35	35,17	2,51	8,71	0,67
53,65	4,18	32,99	2,34	6,47	0,50
52,22	4,01	30,62	2,17	4,29	0,33
50,60	3,85	28,20	2,01	3,24	0,25
48,92	3,68	25,77	1,84	2,18	0,17
47,18	3,51	23,28	1,67	1,24	0,08
45,44	3,34	20,85	1,51	0,75	0,04

Tabla 3.5: Sensor deslizante distancia vs voltaje.

Los anteriores datos se procesaron mediante el *software* MATLAB usando *polyfit* para un ajuste de polinomio de grado 4. En la figura 3.8 se muestra un ajuste bueno de la función resultado, además la gráfica de *residuals*[4] muestra una dispersión aleatoria respecto a *zero line*, validando que el modelo es apropiado para los datos [57].

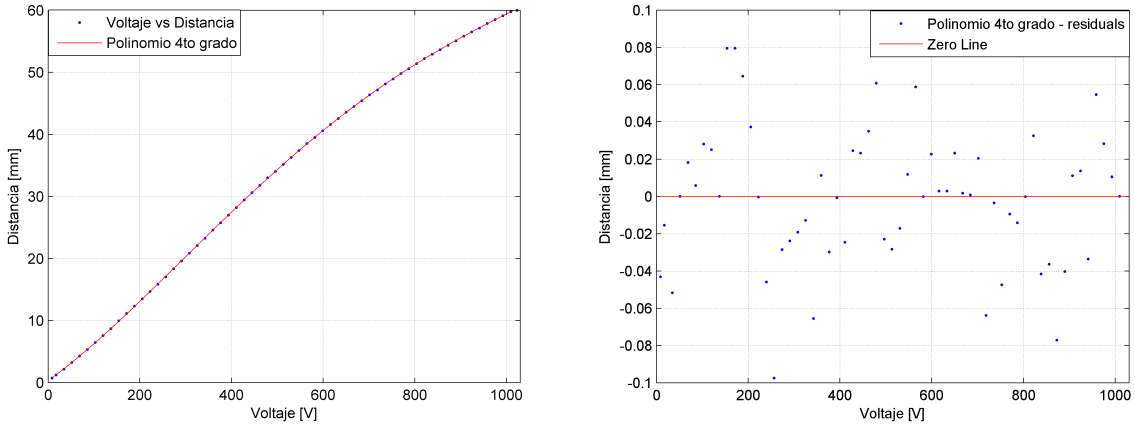

Figura 3.8: Curva experimental voltaje vs distancia potenciómetro lineal.
Fuente: Elaboración propia.

El comportamiento del potenciómetro lineal se representa por un polinomio de 4to orden de la forma $p_1\widehat{x}^4 + p_2\widehat{x}^3 + p_3\widehat{x}^2 + p_4\widehat{x} + p_5$, donde $\widehat{x}$ es normalizado con media μ_1 y desviación estándar μ_2.

$$\widehat{x} = \frac{ADC - \mu_1}{\mu_2}, \{\mu_1 = 513{,}3\ \mu_2 = 303{,}4\}$$

Donde ADC es el valor en cuentas proveniente del Arduino en rango de $0\ a\ 1023$.

Finalmente se extraen los coeficientes del polinomio característico, dando como resultado la ecuación 3.4 que describe el comportamiento del potenciómetro lineal.

$$Distancia\ [mm] = \frac{393}{1189} \cdot \widehat{x}^{\,4} - \frac{553}{765} \cdot \widehat{x}^{\,3} - \frac{3437}{1303} \cdot \widehat{x}^{\,2} + \frac{7867}{397} \cdot \widehat{x} + \frac{36819}{1046} \tag{3.4}$$

[4] *Residual*: Diferencia entre el valor observado y el valor previsto. Cada dato tiene un residuo de la forma:

$$\text{Residual} = \text{valor observado} - \text{valor previsto} e = y - \hat{y}$$

Ejes de referencia de medición

En la ecuación 3.3 se debe tener en cuenta el punto de referencia desde el cual el sensor entrega el valor de posición, y cuál es su sentido de giro. Los puntos de referencia de las articulaciones de los robots se muestran en la figura 3.9, los cuales han sido tomados de acuerdo al MGI del programa de C++.

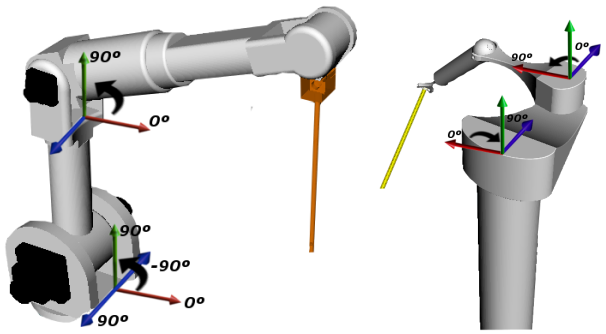

Figura 3.9: Posición de referencia de Hibou y Lapbot.
Fuente: Elaboración propia.

Teniendo en cuenta estas referencias y usando la ecuación 3.3 se realiza el cálculo de la posición de cada articulación restando o sumando el respectivo desfase dependiendo del punto de referencia a medir.

Por ejemplo el potenciómetro se establece en un punto donde su valor de medición sea 1VDC y este punto corresponda con uno de sus ejes de referencia. Si este punto es el valor de referencia de 0° por ejemplo, equivale a restar 90° por 1VDC. Si se tuviera 2VDC el desfase correspondería a 180° y así sucesivamente.

Todas las articulaciones tiene un valor entre 1 y 2 VDC, los cuales permiten asegurar que el voltaje a la entrada no se encuentre por fuera del rango de 0 a 5 VDC correspondiente al ADC.

Mediana de los datos

La función *AnalogRead()* lee el voltaje y convierte esta medición en un número entre 0 y 1023. Los datos técnicos para esta medición especificados para el Arduino son:

- Resolución de 10 bits (0 - 1023 en decimal).
- Exactitud absoluta = +- 2 LSB.
- Tiempo de lectura = Entre 13 y 260 μs

Estas especificaciones indican un error máximo de 2 bits (4 en decimal) para 10 bits, entonces la precisión real máxima del conversor es 4/1023. Además las mediciones se realizan respecto a un voltaje de referencia por *"default"* de 5V, voltaje de polarización del Arduino mediante USB con el computador. La precisión de este voltaje es muy baja, ya que depende de la alimentación del computador y puede llegar a indicar variaciones entre 4.4V y 5.25V dependiendo de la carga energética del computador (periféricos, uso de CPU, tarjeta gráfica, fuente de alimentación, etc.)[58].

Para mitigar o bien reducir variaciones se utiliza una fuente de voltaje independiente para los sensores con el fin de evitar ruido y variaciones de voltaje provenientes del sistema de potencia, además se implementan filtros capacitivos a la salida de la fuente y un acople de tierras entre la fuente de los sensores y el Arduino. Estas modificaciones si bien mejoran la medición, no impiden que se puedan seguir presentando pequeñas variaciones en la lectura del ADC, problema que es físico del *hardware* del Arduino.

El método de la mediana, es una forma de procesar una muestra de datos con el fin de proporcionar datos más estables (ver figura 3.10). Este método toma el valor medio de los datos presentes en la muestra, ignorando los valores extremos por encima o debajo de este, evitando cambios bruscos o aleatorios en algún valor de la muestra[59], lo que desviaría el valor a encontrar, caso que sí sucede cuando se usa un promedio de los datos. Para este caso el método de la mediana se aplica a lotes de 20 datos.

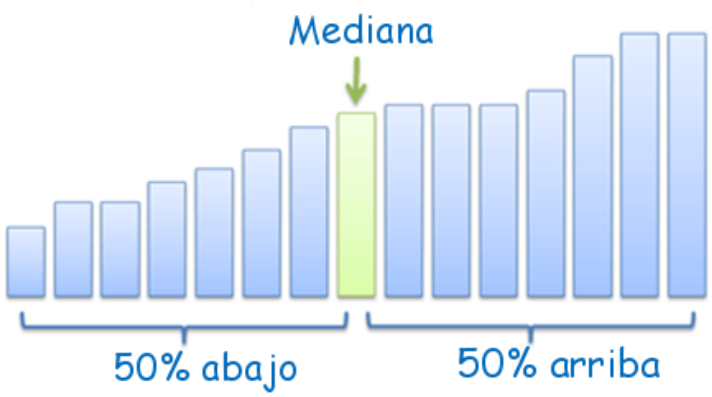

Figura 3.10: Valor de la mediana en una muestra de datos.
Fuente: Elaboración propia.

3.3.6. Ley de control

La ley de control se ejecuta en la función *Control_Law*, la cual recibe el tiempo de muestreo, la referencia y la posición de la articulación en radianes y genera la señal de la ley de control. Este valor de torque [N.m] se encuentra limitado en un rango de -14 a 14, los cuales representan los valores extremos de torque que pueden alcanzar los motores rotativos.

```
float Control_Law ( float Ref , float Output , double Ts)
```

Para las oscilaciones que se pueden tener en la medición, se utiliza una banda muerta de $\pm 1^o$, permitiendo minimizar los efectos de estas variaciones. La banda muerta permite que el control no actúe cuando la señal de error se encuentra dentro del rango determinado, se debe tener en cuenta que una banda muerta grande ocasionaría perdida en la precisión.

Derivator(Error, Ts) deriva la señal de error de posición y obtiene el error de velocidad, necesario para realizar la acción de control según la ecuación 3.1.

```
Ud          = Derivator(Error,Ts);
```

Como se explicó en la sección [3.1] no se utiliza un derivador puro, por el contrario se implementa el modelo de un derivador con un polo el cual permite limitar las ganancias que pueden presentarse por señales de alta frecuencia, es decir señales de ruido.

La función de transferencia del derivador en la ecuación 3.2 y su correspondiente modelo de espacio de estados es[52]:

$$\begin{aligned} U_d(s) &= \frac{1}{\alpha}\left[1 - \frac{1}{\alpha T_d + 1}E(s)\right] \\ &= \frac{1}{\alpha}\left[E(s) - X(s)\right] \end{aligned}$$

Siendo:

$$X(s) = \frac{1}{\alpha T_d s + 1}E(s)$$

La transformada inversa para $X(s)$:

$$\dot{x}(t) = -\frac{1}{\alpha T_d}x(t) + \frac{1}{\alpha T_d}e(t)$$

La transformada inversa para $U_d(s)$:

$$u_d(t) = -\frac{1}{\alpha}x(t) + \frac{1}{\alpha}e(t)$$

Las dos ecuaciones anteriores conforman el modelo en espacio de estados del derivador.

A continuación se procede con el método de discretización, para este caso dado:

$$\dot{x} = f(t, x), \quad x \in R^n$$

Según el método explícito de Euler:

$$x_{k+1} = x_x + hf(t_k, x_k)$$

Si el sistema en tiempo continuo es estable, la selección de h debe garantizar que el sistema en tiempo discreto también lo sea.

Dado:

$$\dot{x} = -\frac{1}{\alpha T_d}x + \frac{1}{\alpha T_d}u, \quad x \in R^n$$

Para el método explícito de Euler:

$$x_{k+1} = \left[1 - \frac{h}{\alpha T_d}\right] x_k + \frac{h}{\alpha T_d}u_k$$

El sistema de tiempo discreto es estable si:

$$0 < h < 2\alpha T_d$$

La realización mínima se muestra en el siguiente fragmento de código, donde T_s indica de tiempo de muestreo.

```
float Derivator(float E, double Ts) {
  static float Xd = 0;
  float U, Td = 1, Alpha = 0.01, C = Ts/(Alpha*Td);
  // State space model
  U      = (1/Alpha)*(-Xd + E);
  Xd     = (1-C)*Xd + C*E;
  return U;
}
```

3.3.7. Escritura de la ley de control

Activación de los motores

Para esta acción, se utiliza un pequeño algoritmo que permite decidir en qué dirección el motor debe moverse.

```
if      ( Control>0.0 ){  opcion = 1;}
else if ( Control<0.0 ){  opcion = 2;}
else opcion = 3;

   switch(opcion){
   case 1:
    digitalWrite (DirM_1, HIGH );
    digitalWrite (DirM_2, LOW );
    analogWrite: (PwmM, Control_PWM);
   break;
   case 2:
    digitalWrite (DirM_2, HIGH );
    digitalWrite (DirM_1, LOW );
    analogWrite: (PwmM,Control_PWM);
   break;
case 3:
    digitalWrite (DirM_1, LOW );
    digitalWrite (DirM_2, LOW );
    analogWrite: (PwmM, Pwm_cero);
   break;
   }
```

El signo de la variable *control* permite saber el sentido de giro del motor. El puente H funciona con 2 entradas digitales las cuales permiten definir el sentido del giro (ver tabla 3.6).

IN1	**IN2**	**PWM**	**MOTOR**
1	0	Active	Forward
0	1	Active	Backward
0	0	Active	Brake
x	x	Low	Off

Tabla 3.6: Manejo puente H HMD-02.

Limitación del PWM por articulación

La variable *Control_PWM* que contiene el ciclo de trabajo del PWM, está limitada de acuerdo a la articulación a controlar. Debido a la baja velocidad con la que se deben mover los motores se evita usar rango completo del PWM. Tal como se explicó en la sección 3.3.3 existen valores bajos de PWM para los cuales los motores no logran arrancar, y valores muy altos que generarían una velocidad demasiado alta. El rango utilizado para control del PWM para las articulaciones es entre 10 y 80 [5], dependiendo del torque necesario para cada articulación.

[5]La señal PWM en Arduino comprende valores en un rango entre 0 y 255. Este valor es adimensional.

Capítulo 4

Software Robotic Environment for Laparoscopic Surgery (RELAS)

RELAS for Hibou and Lapbot es el nombre otorgado al software utilizado para la manipulación virtual de los robots Hibou y Lapbot. El software permite comunicarse con la nueva tarjeta de adquisición de datos por medio del protocolo de comunicación serial, ofreciendo información en tiempo real [1] de las posiciones articulares de los robots y sus errores articulares y cartesianos. El software permite visualizar los movimientos reales de los robots. Los movimientos de los robots están dados por consignas geométricas definidas o por un dispositivo externo denominado *joystick*, a partir del cual se obtienen las posiciones cartesianas X, Y y Z deseadas para los robots.

4.1. Descripción general del software

El desarrollo de RELAS para los robots Hibou y Lapbot tiene como punto de partida el ambiente virtual del simulador quirúrgico realizado por Guerrero y Méndez[60] en una asignatura de la Maestría en Automática de la Universidad del Cauca. El simulador base consta de una habitación como espacio de trabajo, en la cual los robots Hibou y Lapbot se encuentran posicionados en lados contrarios de una camilla. Sobre la camilla hay una colchoneta y una caja de dimensiones definidas en [32], con la cual se simula la cavidad abdominal y es usada como espacio operatorio. La caja cuenta con dos trocares que se utilizan como puntos de incisión y que sirven para verificar que se respeta el paso por el trocar.

[1] El tiempo real hace referencia a 10ms utilizados como tiempo de muestreo

Las herramientas software que se utilizaron durante el desarrollo del simulador base y que se mantienen para el desarrollo de este proyecto son *Microsoft Visual Studio 2008* como entorno integrado de desarrollo que permite realizar programación orientada a objetos integrando librerías como Qt y motores gráficos tales como VTK. Qt es una biblioteca multiplataforma usada para desarrollar aplicaciones con interfaz gráfica de usuario, para este caso se utilizan las librerías en su versión 4.7.4. Qt incluye un IDE de creación de interfaces gráficas llamado Qt Creator con el cual se realizó el menú principal del software RELAS, la versión utilizada fue Qt Creator 2.3.0. La realización de gráficos 3D por computadora, procesamiento y visualización de imagen se lleva a cabo con el motor gráfico VTK en su versión 5.8.0.

El software RELAS permite mover los robots a partir de consignas cartesianas geométricas predeterminadas o mediante las posiciones cartesianas X, Y, y Z provenientes de un *joystick*, las posiciones cartesianas del *joystick* son vistas en el efector final. Estas consignas pasan por el modelo geométrico inverso para generar las posiciones articulares deseadas para cada articulación. Las posiciones articulares deseadas se transmiten a la tarjeta electrónica utilizando el protocolo de comunicación serial. En la tarjeta electrónica se realiza el control de los robots utilizando un controlador PID paralelo; los datos de las posiciones articulares reales captados por los sensores son enviados desde la tarjeta electrónica al software RELAS en el cual se procesan y se muestran en las gráficas de errores, donde adicionalmente se usan como realimentación para mostrar gráficamente el movimiento real de los robots.

En la barra de menús se encuentran las opciones para inicializar la comunicación serial, leer las posiciones articulares, leer la posicion cartesiana del *joystick*, graficar los errores articulares y error cartesiano en ventanas independientes. Una representación básica de la comunicación entre los dispositivos externos con el *software* RELAS se aprecia en la figura 4.1.

Las funciones principales del nuevo software son:

- Permite la interacción entre los dos robots (Hibou, Lapbot) dentro de la cavidad abdominal, a través de un dispositivo de mando (*joystick* o *gamepad*).
- Cuenta con una interfaz gráfica de usuario amigable permitiendo que la aplicación sea de fácil uso.
- Muestra diferentes gráficas y datos sobre la posición y error actual de los movimientos de los robots.
- Muestra menús de configuración y manejo de la comunicación serial.

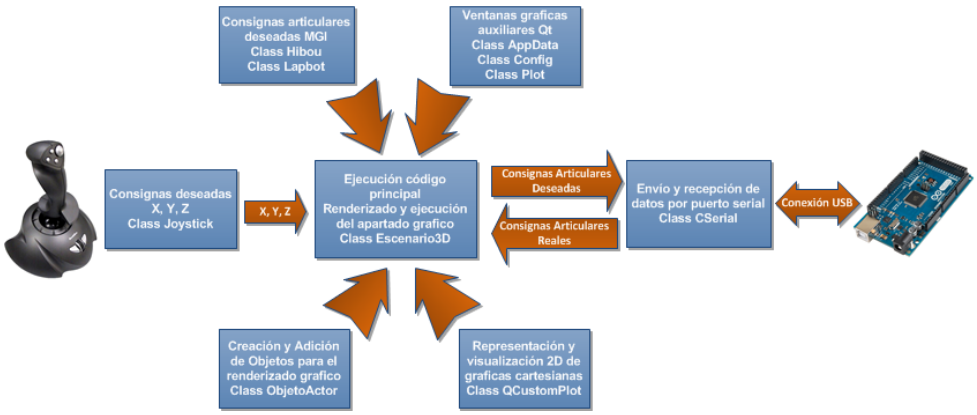

Figura 4.1: Esquema general de funcionamiento de RELAS.
Fuente: Elaboración propia.

4.2. Desarrollo del software

Durante el desarrollo de un software e incluso en la vida cotidiana se plantean objetivos y metas las cuales se quieren llevar a cabo satisfactoriamente, sin embargo uno de las interrogantes que se presentan desde un principio es: ¿cuál es mejor forma de realizarlo?. Si bien no existe una respuesta absoluta a este interrogante, la ingeniería del software expone diversas metodologías que ayudan a realizar un software de la mejor forma posible. En el desarrollo de este software no se aplica ninguna metodologia de forma completa, pero si se utilizan herramientas de modelado para ayudar a la organizacion y documentacion. Como paso inicial se exponen algunos conceptos necesarios en el desarrollo de software.

Un software, así como cualquier producto, tiene un ciclo de vida comprendido desde el momento en que surge la idea, hasta aquel momento en que deja de ser utilizado por el último de sus usuarios. Este ciclo de vida varía según el objetivo, la población a la cual va dirigida, su forma de construcción, la robustez del desarrollo y el presupuesto, entre otros. La forma natural en que se desarrolla un software, parte desde una necesidad hasta llegar a la puesta en marcha de una solución y su apropiado mantenimiento. Las etapas que se utilizan para un ciclo de vida son las siguientes:

- Planeación: Se consideran las condiciones iniciales del software.
- Desarrollo: Se realiza el diseño, implementación y validación del software basado en los requisitos del mismo.
- Operación: Se hace entrega el software para su producción y uso por parte del usuario.

- Mantenimiento y evolución: Se hacen corrección de anomalías presentadas en los usuarios y la actualización de componentes del software.

Si bien no se implementa un modelo o metodología propia de la ingeniería del software en este proyecto, se organiza una estrategia para llevar a cabo el desarrollo de RELAS basado en herramientas UML.

4.2.1. Metodología de programación

En el modelo clásico de desarrollo de software *Built & fix* el software se desarrolla sin ninguna especificación o diseño [61]. El producto inicial se construye y luego se modifica varias veces hasta que el software satisface al usuario. Es decir, el software se desarrolla y se entrega al usuario, el usuario comprueba si las funciones deseadas están presentes, si no entonces el software se cambia de acuerdo a las necesidades mediante la adición, modificación o supresión de funciones. Este proceso continúa hasta que el usuario siente que el software puede ser utilizado de manera productiva, sin embargo la falta de requisitos de diseño y modificaciones repetidas resultan en pérdida de aceptabilidad de software y de tiempo.

Para el desarrollo de este proyecto se han utilizado herramientas UML para abarcar las etapas de planeación y diseño de software, luego se pasa a la etapa de codificación y finalmente ejecución. Estas etapas son repetidas cíclicamente una después de la otra con el fin de eliminar todos los problemas que fueron encontrados en los ciclos anteriores.

Esta metodología se lleva a cabo de forma individual para cada uno de los requerimientos del software, los cuales son:

- Realizar la comunicación serial con el dispositivo Arduino atendiendo los eventos de recepción de datos.

- Simular las consignas geométricas en 2D y 3D para verificar la movilidad de los robots.

- Mover los robots de la aplicación con los datos realimentados.

- Realizar el ciclo de renderizado y funcionamiento del software en un tiempo menor a 10ms, con el fin de cumplir con el tiempo de muestreo.

4.2.2. Modelado UML

El modelado de software o cualquier sistema brinda una mayor organización y formalidad a la hora de realizar un desarrollo. El uso de modelos ayuda al desarrollador a visualizar y proyectar las acciones futuras en el sistema a construir. El *"Lenguaje de Modelamiento Unificado"* UML, es un lenguaje gráfico para visualizar, especificar, construir, organizar y documentar cada una de las partes que comprenden el desarrollo de software. UML entrega una forma de modelar conceptualmente las funciones del sistema.

Los casos de uso se utilizan en UML como herramienta para analizar los requerimientos del cliente o requisitos software. A través del modelado de casos de uso, el usuario llamado actor que interactúa con el software, es modelado con las funcionalidades que ellos requieren en el sistema, es decir con los casos de uso.

Casos de uso

Los casos de uso describen qué hace un sistema pero no especifica cómo lo hace, además de ayudar a la comprensión del sistema para desarrolladores futuros y usuarios finales.

Los casos de uso propuestos para el desarrollo del software RELAS se describen en la figura 4.2.

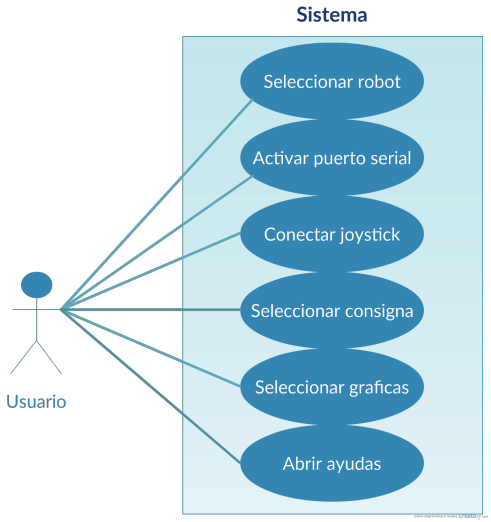

Figura 4.2: Diagrama de casos de uso RELAS.
Fuente: Elaboración propia.

La figura 4.3 describe la arquitectura de los casos de uso en el sistema mostrando sus relaciones entre ellos y el usuario. Las relaciones entre casos de uso llevan una flecha de navegación que apunta a quién inicializa la interacción con el sistema. Si no existe una flecha significa que no hay interacción entre el actor y el caso de uso.

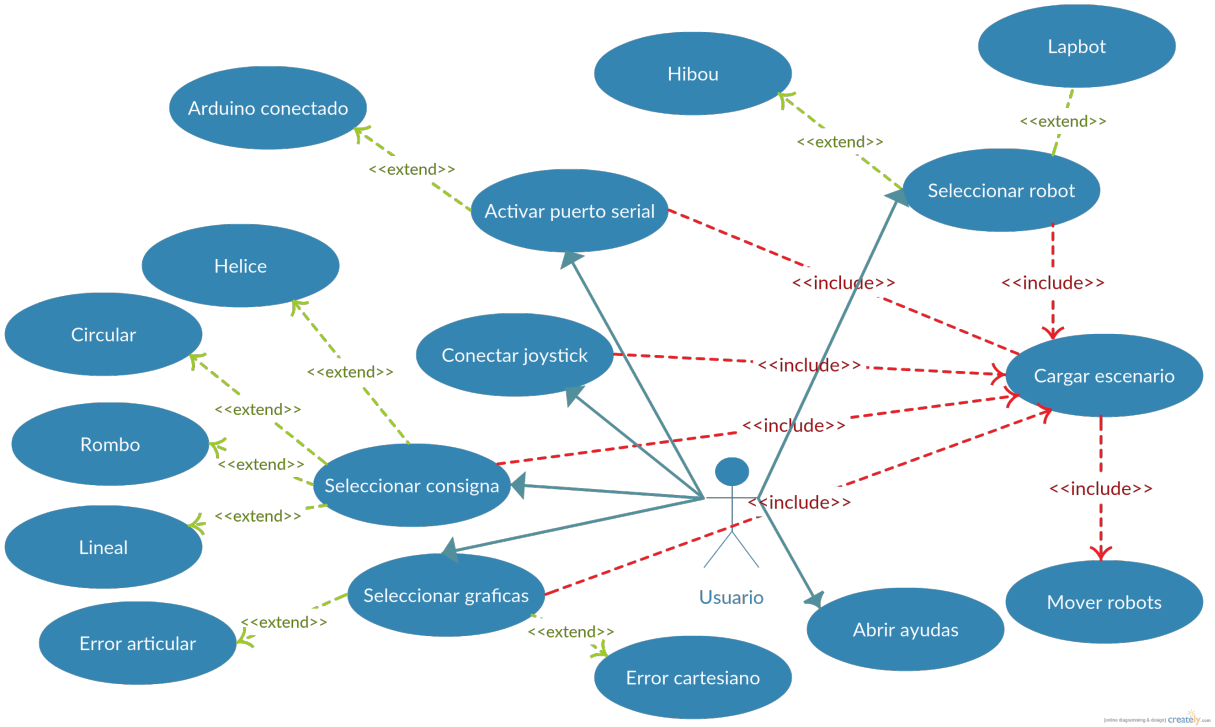

Figura 4.3: Diagrama de interacción de casos de uso de RELAS.
Fuente: Elaboración propia.

Diagrama de Clases

Aunque los diagramas de clase hacen parte del análisis dentro del desarrollo de un software, de igual forma proyectan la forma en que se va a codificar y estructurar el software. Las clases que se modelan son identificadas con sus respectivas relaciones y son descritas en un diagrama de clases (ver figura 4.4).

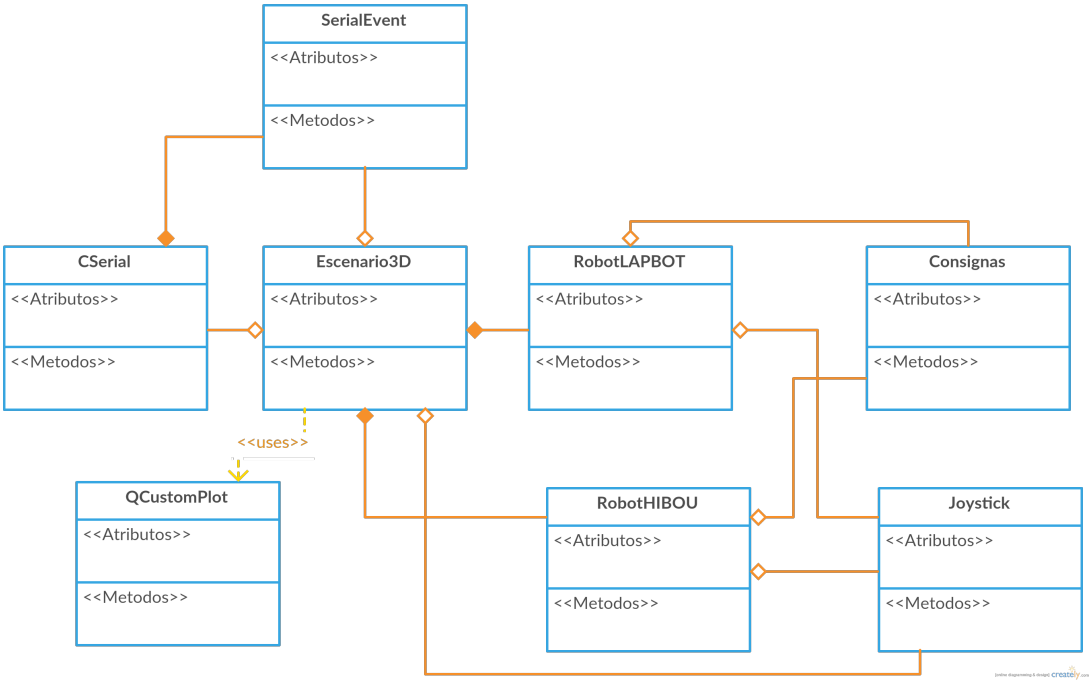

Figura 4.4: Diagrama de clase para RELAS.
Fuente: Elaboración propia.

4.3. Nuevas clases presentes en RELAS

El software base ha sido utilizado en diferentes trabajos de grado previos a este [32, 42–44], por esta razón este trabajo se centra en detallar los elementos que han sido añadidos y/o modificados para llegar a la versión final del software RELAS.

4.3.1. CSerial

Para realizar la comunicación entre el software RELAS y la nueva tarjeta electrónica para envío datos, tal como las consignas deseadas de control, se necesita la adición y configuración de una librería al entorno de *Microsoft Visual C++ 2008*. Existen diferentes librerías en plataformas para desarrolladores de *software* para uso y configuración del protocolo serial, que permiten la comunicación entre un entorno de programación en C++ y un dispositivo serial. Después de diferentes pruebas realizadas sobre diferentes librerías, se elige una de ellas principalmente por contar en su estructura con los elementos necesarios para el funcionamiento del protocolo serial, además de presentar robustez en su programación e incluir manejo y atención de eventos por entrada de datos.

La comunicación serial se realiza utilizando el grupo de librerías (3 en total)[62], para

permitir el manejo el protocolo serial en aplicaciones que usarán la interfaz de programación de aplicaciones de *Windows*, comúnmente llamada API Win32.

Las tres librerías se conectan entre ellas para realizar la comunicación serial, por esta razón se deben incluir todas en el árbol del proyecto. A continuación se explica brevemente las características más destacables de cada una de las librerías.

1. **CSerial:** Es la librería que contiene todos los parámetros y funciones para configurar el protocolo serial, además de permitir el uso del protocolo de forma regular. Las otras dos librerías pueden realizar configuración de los parámetros heredando esta clase.

2. **CSerialEx:** Permite manejar eventos del puerto serial en un hilo independiente. Su dificultad radica en que debe agregarse manejo de arquitectura por hilos en el software en que se desee implementar. El *software* RELAS no ha sido realizado utilizando estructura por hilos, razón por la cual este modo no puede ser utilizado.

3. **CSerialWnd:** Permite manejar eventos del puerto serial por ventana. Cada vez que se presente un evento de comunicación por el puerto serial, permite realizar el llamado de una función predeterminada o mostrar un mensaje.

Las funciones más importantes en las librerías y que son utilizadas para el desarrollo del *software* RELAS son:

- **Open:** Permite abrir el Puerto COM especificado.

- **Setup:** Inicializa la comunicación serial, configurando los parámetros de envío y recepción de datos tales como velocidad de transmisión, bit de paridad, bits de datos, bits de parada.

- **ReadData:** Lee los datos del *buffer* de entrada serial, esta función recibe como argumentos la variable en la cual se almacenarán los datos leídos y la cantidad de bits de datos a leer. Esta variable debe ser de tipo vector de caracteres (char[]).

- **WriteData:** Escribe los datos en el *buffer* de salida serial, esta función recibe como argumentos la variable en la cual se tienen los datos y la cantidad de bits de datos a enviar. Esta variable debe ser de tipo vector de caracteres (char[]).

- **IsOpen:** Verifica la conexión del Arduino al puerto serial. El retorno de esta función es *Verdadero* en caso de que este se encuentre conectado, y *Falso* en caso contrario.

- **Purge:** Permite limpiar el *buffer* de salida y entrada del computador. Esto permite eliminar posibles errores por datos que se encuentren retrasados o almacenados en el *buffer*.

Todas las funciones utilizadas en el desarrollo del software para comunicación serial son llamadas desde la librería **CSerialWnd**.

En aplicaciones donde se requiere alta velocidad de transmisión de datos debido a manejo de información en tiempo real y/o funciones de control que exijan tiempos de muestreo fijos, se debe asegurar que el software se ejecute lo más rápido posible sin generar retardos de tiempo, bucles o tiempos de espera en la aplicación. Se hace uso de CSerialWnd por su importante característica de atender eventos serial por entrada de datos, la cual permite generar una señal de activación para atender eventos y así evitar bucles para recepción de datos y tiempos de espera para recibirlos, bloqueando el *software* causando variaciones en el periodo de muestreo o las funciones normales del *software*.

Evento serial

Por defecto, la función *OnEvent* definida en la clase *CSerialWnd* se ejecuta automáticamente cuando un evento es detectado en la comunicación serial. En esta función se puede hacer directamente la lectura y almacenamiento de los datos, pero estos no estarían visibles para la clase principal Escenario3D.
Para resolver este problema se hace uso de *Signals & Slots*, mecanismo propio de QT para comunicación entre objetos, que permite generar una señal cuando *OnEvent* se llame a atender el evento de entrada de datos. Dado que la clase *CSerialWnd* no es un miembro Q_OBJECT[2] no puede generar una señal de este tipo, por tanto se crea una nueva librería llamada *SerialEvent* que genera una señal que indica la entrada de datos. La linea 6 del siguiente código se llama la función *SetEvent()* , la cual genera la señal de aviso de entrada de datos.

[2]QObject es el corazón del modelo de objetos de Qt y constituye la clase base para muchas clases importantes en las librerías de QT. Se puede conectar una *signal* a un *slot* con *connect ()* y destruir la relación con *disconnect()*.

```
1 void CSerialWnd::OnEvent (EEvent eEvent, EError eError) {
2 //Reset and Save event
3  int  Evento1= GetEventType();
4  eEvent = EEvent();
5 //Llamada a la nueva clase SerialEvent
6  FlagEvent.SetEvent();          }
```

OnEvent llama a la función *SetEvent* perteneciente a la nueva clase creada, en la cual se implementa una linea de código que emite la señal de aviso de entrada de datos a una función en *Escenario3D*.

```
1 void SerialEvent::SetEvent() {     emit EmitSerialEvent(); }
```

Es de resaltar que la señal es conectada desde *SerialEvento*, objeto o instancia[3] creado para la clase *CSerialWnd*. A su vez FlagEvent es un objeto de la clase *SerialEvent* que está definido dentro la clase *CSerialWnd*, es decir que este pertenece al objeto *SerialEvento*. Por tanto para poder acceder al objeto *FlagEvent*, se debe hacer a través del objeto propietario, en este caso *SerialEvento*. En el siguiente código, en las lineas 3 y 4, se muestran las definiciones de las instancias y finalmente en la linea 7 la señal es conectada a la función *EventoSerialRead* en la clase principal *Escenario3D*, donde se realiza la lectura, almacenamiento y procesamiento de los datos provenientes de la nueva tarjeta electrónica.

```
1 //Creacion de instancias
2 public:
3  CSerialWnd SerialEvento; //Class CSerialWnd
4  SerialEvent FlagEvent; //Class  SerialEvent
5 //La señal se genera desde la Instancia SerialEvent(class SerialEvent) creada en SerialEvento(
      instancia de CSerialWnd)
6 //Conexión para el Evento Serial – Lectura de los datos al recibir la señal de llegada de datos
7  connect( &SerialEvento.FlagEvent, SIGNAL(EmitSerialEvent()), SLOT(EventoSerialRead()));
```

Configuración e inicio de CSerial

La librería *SerialWnd* debe configurarse para realizar la detección de eventos dado que estos no están activados por defecto. La función void $Escenario3D$::$actionIniciar\_triggered()$

[3]Una clase es la definición de un objeto de programación, y una instancia de una clase es justamente el objeto generado a partir de esa clase. Puede haber muchas instancias de una clase, o en otras palabras, varios objetos generados a partir de esa clase.

configura e inicia el puerto serial según los parámetros necesarios de este *software*.

En la línea 2 del siguiente código se inicia el puerto serial con el valor del puerto COM configurado en la variable *Serial_Com*. Los otros parámetros son necesarios para configurar la clase pero no son relevantes.

En la línea 4 se configura la velocidad y el número de datos de envío y recepción. Se configura el puerto COM a una velocidad de 115200 baudios, 8 bits de datos, sin bit de paridad y 1 bit de *stop*.

En la línea 6 se configura el evento *EEventRcvEv*, el cual permite generar el evento de recepción serial. El evento serial puede ser configurado para activarse por diferentes sucesos, para este caso se configura para que el evento se active solo cuando un carácter predefinido sea recibido en el *buffer* serial. Se usa "!" como carácter de activación del evento, este a su vez se usa como detector de errores por ser el último en la trama de datos, indicando que los datos han sido recibidos correctamente.

El puerto serial puede ser configurado es dos modos de operación: *EReadTimeoutNonblocking* lee solo los datos que se encuentren disponibles en el *buffer* del puerto serial y *EReadTimeoutBlocking* se usa cuando se conoce el número exacto de bits que van a ser leídos, y además permite usar solo un proceso de lectura para almacenar todos los datos, caso contrario al primer modo, donde debe usarse varias sentencias de lectura para almacenar toda la trama.

Dado que se conoce la longitud de la trama de datos que siempre será recibida, este último modo es configurado en la línea 11. Finalmente en la línea 12 se envía un carácter $, el cual se usa como indicador para que Arduino inicie la trasmisión de datos, de lo contrario este permanecerá a la espera sin realizar acciones de control o lectura de los sensores.

```
1  // Inicializaci ón del Puerto Serial
2   SerialEvento.Open(_T(Serial_Com),hwndDest,WM_NULL,lParam,0,0);
3  //Configuración del Puerto Serial
4   SerialEvento.Setup(CSerial::EBaud115200,CSerial::EData8,CSerial::EParNone,CSerial::EStop1);
5  // Register only for the receive event
6   SerialEvento.SetMask(CSerial::EEventRcvEv);
7  //Declaración de Variable Inicial en la Cadena de Caracteres de Recepción
8   char bEventChar[2] = {"!"} ;   bool fAdjustMask = true;
9   SerialEvento.SetEventChar(bEventChar[0], fAdjustMask);
10 // Use 'blocking' reads, because we know how many bytes will be received.
11  SerialEvento.SetupReadTimeouts(CSerial::EReadTimeoutBlocking);
12  SerialEvento.Write("$");
```

Lectura de datos

La función *void Escenario3D::EventoSerialRead()* se inicia únicamente cuando el evento de recepción de datos es activado. En la línea 2 del siguiente código, se realiza la lectura de datos y se almacenan en la variable *xRchar*, posteriormente se realiza la separación de las variables de acuerdo a los identificadores presentes en la trama y se guardan en unas nuevas variables para su manejo dentro del programa principal, para la primer articulación en la linea 5 se restan 300 de acuerdo a lo explicado en la sección 3.3.4 . En la línea 10 se calculan las posiciones X, Y y Z con el MDG para la obtención del error cartesiano.

Finalmente se llama la función *EventoSerialWrite* como parte del sistema **Llamada-Respuesta**[4], el cual tiene como prioridad enviar las nuevas posiciones deseadas.

```
1  //Lecura de datos
2   SerialEvento.Read(xRchar,sizeof(xRchar));
3  //Almacenamiento en variables principales (Hibou)
4   HJ1 = Join_int[0]-300; //Desfase para Art1
5   HJ2 = Join_int[1];
6   HJ3 = Join_int[2];
7  //Calculo del error cartesiano
8   Error_CartH = Diferencia(xE,yE,zE, Hibou.HPx, Hibou.HPy, Hibou.HPz);// X,Y,Z
9  //Llamada a la función de envío de datos para enviar las consignas por el puerto serial
10  EventoSerialWrite();
```

Escritura de datos

La función *void Escenario3D::EventoSerialWrite()* genera una nueva cadena de datos con las consignas deseadas, añadiendo los caracteres de diferenciación de articulaciones. En esta misma función se almacenan todas las consignas enviadas para ser comparadas y verificar los errores obtenidos.

```
1   string   Ast = "*";
2   string   Num = "#";
3   string   Por = " %";
4   string   Ad  = "!";
5  std::string  AllInOne;
```

[4]Permite enviar una respuesta de datos al recibir la llamada del evento. Dado que Arduino envía datos cada 10 ms, la lectura de datos se realiza en este mismo periodo de tiempo, e inmediatamente se envían los nuevos datos, conservando este tiempo de muestreo

Los datos son enviados de acuerdo al robot que se encuentre seleccionado en el software, en el caso del siguiente código, para Hibou.

```
 if (SHibou){ //robot Hibou activado
AllInOne = Ast + (to_string(HJ1D+300))+ Num + (to_string(HJ2D))...
       + Por + (to_string(HJ3D)) + Ad ;
      }
//Envio de nuevos datos
 const char *AllInOneChar = AllInOne.c_str(); //Convierto de String a Char para ser enviados
 SerialEvento.Write(AllInOneChar);
```

4.3.2. QCustomplot

Con el fin de proveer una interfaz de visualización de datos clara, configurable y eficaz, se hace uso de una robusta librería para la representación, trazado y visualización de gráficas 2D llamada QCustomPlot [63]. Esta es un *widget* de Qt basado en C++ que cuenta con una amplia documentación para su manejo y posee un alto rendimiento en aplicaciones de tiempo real. Además permite exportar los gráficos a PDF o archivos de imagen como JPG y PNG.

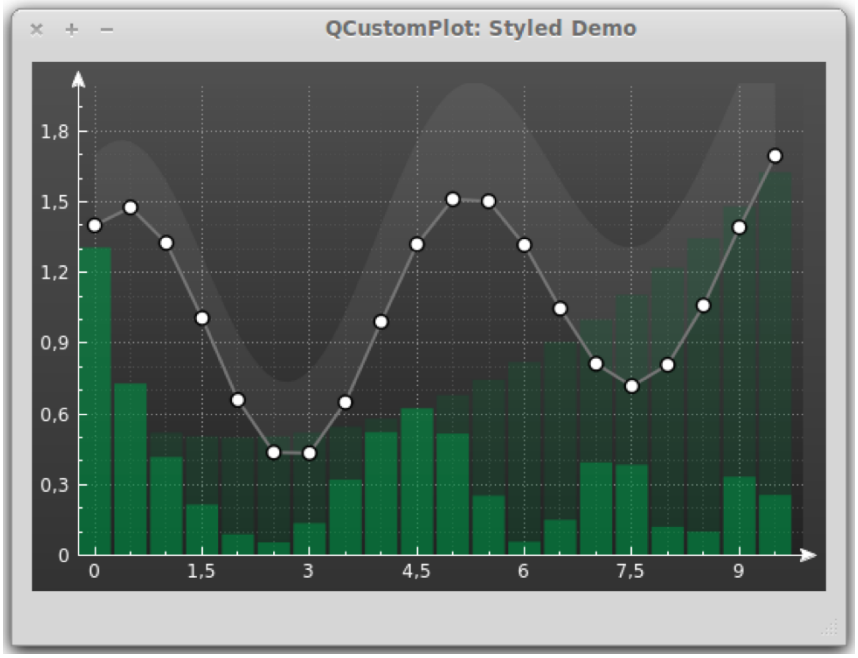

Figura 4.5: Gráfica de muestra QCustomplot.
Fuente: Tomada de [63].

Declaracion y manejo

Para su manejo se debe crear una ventana visual en QT, y dentro de ella generar un QWidget, donde debe ser llamado QCustomPlot. Para el manejo de esta aplicación esta nueva ventana gráfica recibe por nombre *Plot*[63].

Gráficas principales

Cada gráfica es creada por una instancia diferente de *Plot* como se ve en el siguiente código, esto permite realizar gestión de datos, configuración visual de las ventanas emergentes de forma independiente para todas las gráficas.

```
//Instancia ventana de gráficas
 PlotH = new Plot(this); //Hibou Articular
 PlotL = new Plot(this); //Lapbot Articular
 PlotH_C = new Plot(this); //Hibou Cartesiano
 PlotL_C = new Plot(this); //Lapbot Cartesiano
 PlotConsigna = new Plot2(this); //Consignas Cartesianas XY
```

La actualización de valores para cada gráfica se realiza en la función *realtimeDataSlot*, mostrada a continuación, que se llama con un *timer*, el cual realiza la llamada tan pronto como el sistema termine de atender todas las tareas pendientes.

```
 //Conexión del Timer con la función de actualización
 connect(dataTimer, SIGNAL(timeout()), this, SLOT(realtimeDataSlot()));
 dataTimer->start(0); // Interval 0 means to refresh as fast as possible
//Adicion de datos a las gráficas (Error cartesiano)
 double Hvalue0 = ((HJ1D - HJ1)*0.1); //Art 1 Deseados -Reales
 double Hvalue1 = ((HJ2D - HJ2)*0.1); //Art 2
 double Hvalue2 = ((HJ3D - HJ3)*0.1); //Art 3
//Add data to lines:
 PlotH->ui->customPlot->graph(0)->addData(key, Hvalue0);
 PlotH->ui->customPlot->graph(1)->addData(key, Hvalue1);
 PlotH->ui->customPlot->graph(2)->addData(key, Hvalue2);
```

La configuración de todas las ventanas, títulos, etiquetas, rangos de trabajo y configuración de colores se encuentran en la función *void Escenario3D::Inicializar_Graficas()*. Una descripción completa de los códigos utilizados se encuentran en la documentación online de QCustomPlot [63] y los ejemplos que se presentan.

4.3.3. Graficación en segundo plano usando hilos

Dado que la actualización y renderizado de los gráficos en VTK podría llegar a emplear un consumo de tiempo excesivo de acuerdo a las especificaciones, sumado que la adición y actualización de las nuevas ventanas gráficas generaban un retardo de tiempo considerable que impedía que la aplicación lograra cumplir el tiempo de muestreo, se considera implementar un segundo hilo que se encargue del manejo de las gráficas.

Para solucionar este inconveniente la función *ReplotGraphs* es ejecutada en un segundo hilo usando la característica de Qt *QtConcurrent::run*, la cual permite ejecutar funciones del programa en un hilo independiente [64].

```
1 void Escenario3D::ThreadPlot(){
2  //Ejecuta la Función ReplotGraphs en un segundo Hilo.
3  if(ReplotEnd){
4   ReplotEnd=false; //Bandera -> Evitará que se llame de nuevo la función de actualización cuando
        esta no se ha completado totalmente
5   QtConcurrent::run(this,&Escenario3D::ReplotGraphs);
6   }
7 }
```

void ReplotGraphs se encarga de remover datos antiguos, escalar los ejes X y Y, y actualizar la ventana para todas las instancias gráficas presentes en el software, estas acciones pueden apreciarse en el cuadro de código siguiente.

```
1 //RangeTime = segundos de rango
2 // remove data of lines that's outside visible range:
3  PlotH-> ui->customPlot->graph(0)->removeDataBefore(key-RangeTime);
4  PlotH-> ui->customPlot->graph(1)->removeDataBefore(key-RangeTime);
5  PlotH-> ui->customPlot->graph(2)->removeDataBefore(key-RangeTime);
6 // rescale value (vertical) axis to fit the current data:
7  PlotH->ui->customPlot->rescaleAxes();
8 // make key axis range scroll with the data at a constant range size of RangeTime = segundos de
      rango
9  PlotH->ui->customPlot->xAxis->setRange(key+0.25, RangeTime_H, Qt::AlignRight);
10 //Actualiza ventana gráfica
11  PlotH->ui->customPlot->replot();
```

4.4. Funcionalidades y uso del software

En *RELAS* se implementaron diferentes menús para acceder a las funciones requeridas por el usuario. Los menús permiten iniciar o detener la comunicación serial, abrir las gráficas de error o iniciar una consigna predefinida (ver figura 4.6).

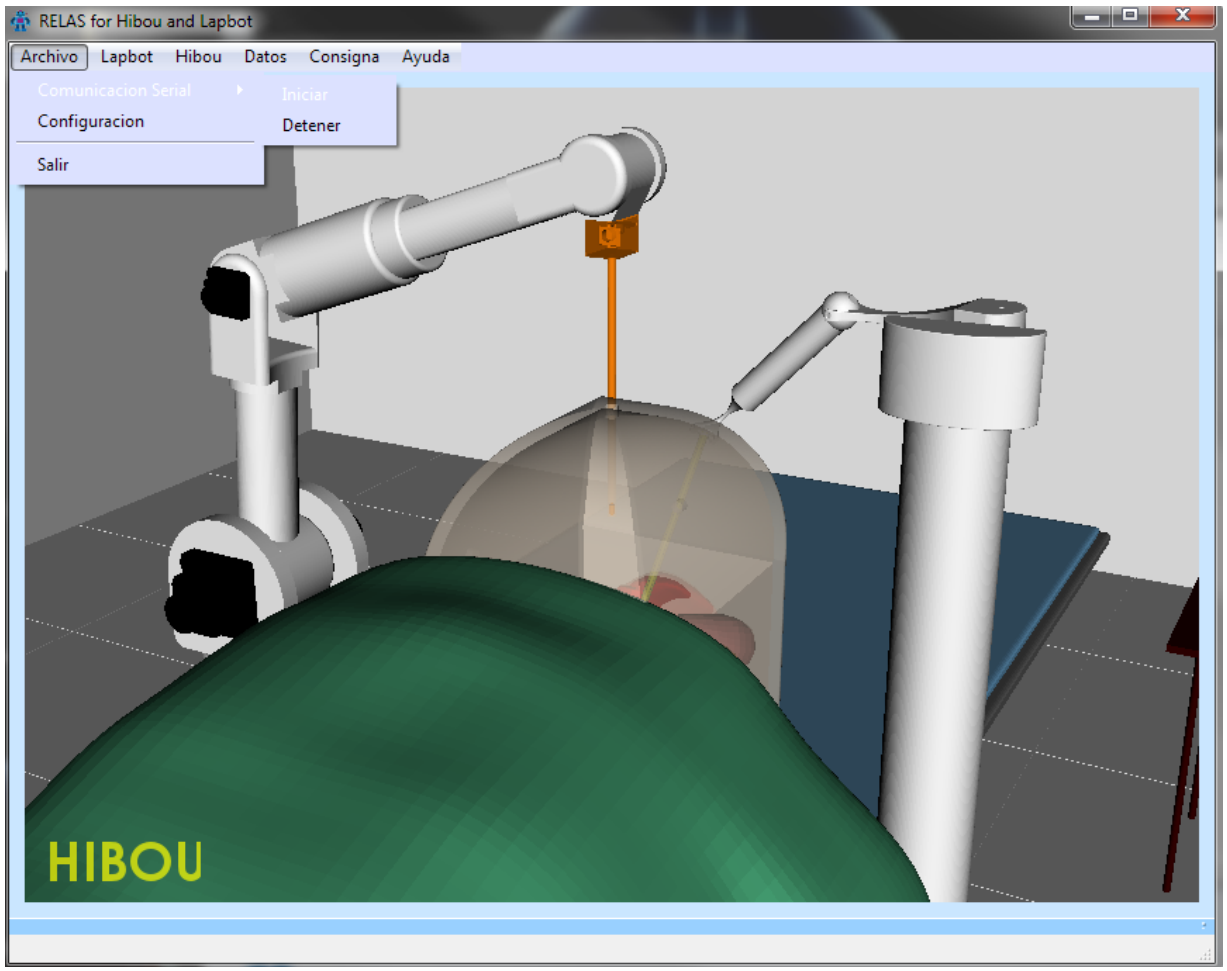

Figura 4.6: Ventana principal RELAS con menús.
Fuente: Elaboración propia.

Se cuenta con 5 diferentes consignas: circular, lineal, elipse, rombo y 3D. Las dimensiones y tiempo de estas consignas son predefinidos y no puede ser cambiados desde la interfaz visual, por ejemplo en la figura 4.7 se ve la consigna rombo para el robot Hibou.

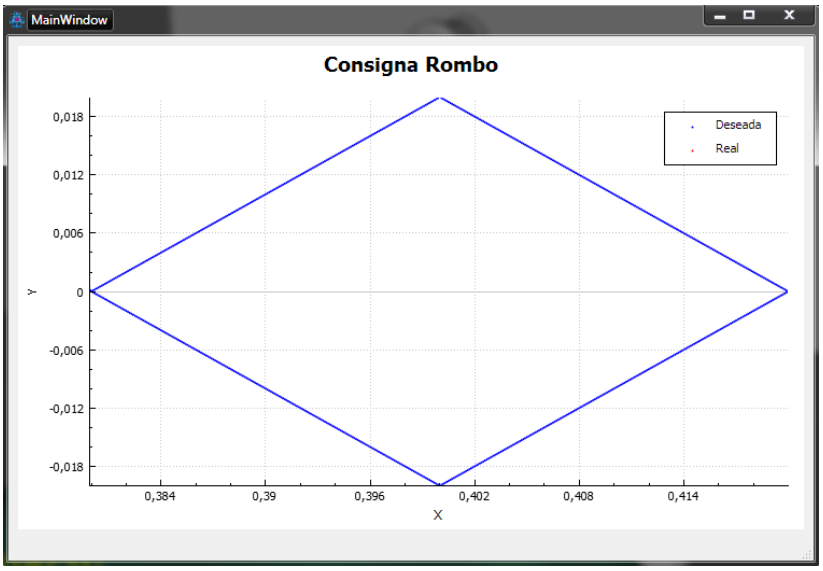

Figura 4.7: Consigna rombo.
Fuente: Elaboración propia.

Las gráficas de error cartesiano o error articular permiten ser pausadas o editadas en su rango de tiempo. Estas gráficas se escalaran automáticamente en el eje Y según los datos presentes en el rango de tiempo establecido (ver figura 4.8).

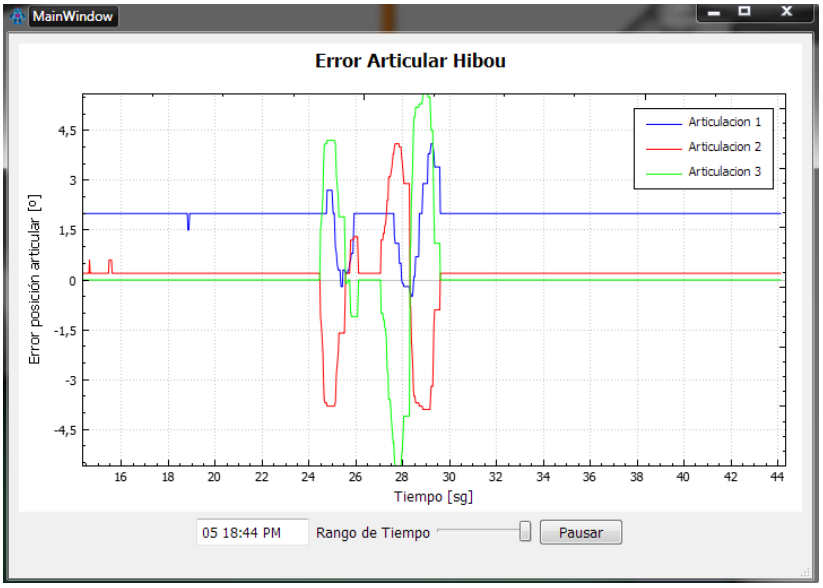

Figura 4.8: Error articular Hibou.
Fuente: Elaboración propia.

Una completa descripción de todas las funcionalidades, mensajes, restricciones y cómo usar el software se encuentra en el Anexo A.

4.5. Resultados - Consigna 3D

Como finalización del capítulo se presentan los resultados obtenidos para la consigna 3D realizada por los prototipos reales de los robots Hibou y Lapbot.

4.5.1. Hibou

Las consignas realizadas por Hibou y Lapbot tienen las mismas características, la consigna realizada por Hibou tiene centro en $x = 40$ cm, $y = 0$ cm y un radio equivalente a 2 cm. La distancia recorrida en z es de 4cm teniendo como punto de inicio una distancia de 11 cm y finalizando en 15 cm. El tiempo de realización de la consigna fue de 10 segundos (ver figura 4.9).

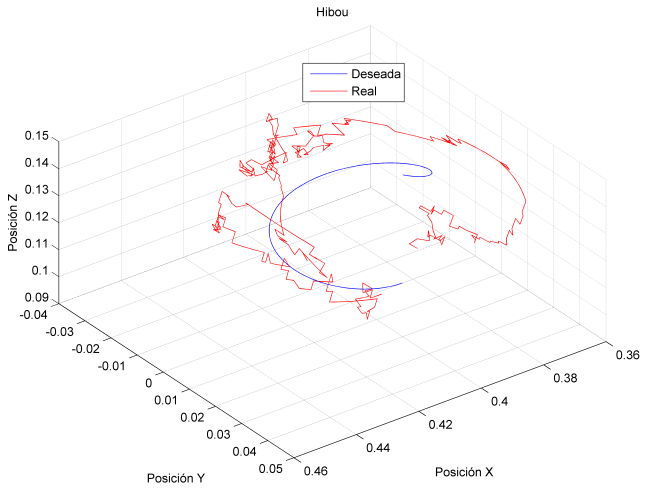

Figura 4.9: Consigna 3D realizada por el robot real Hibou.
Fuente: Elaboración propia.

Se observa que el robot Hibou no fue capaz de seguir la referencia en el eje z, debido a que dos de sus tres articulaciones tienen movimientos sobre este eje, por ende los movimientos en su gran mayoría afectan al eje Z. Debido a la construcción física, peso y tipo de motores no se logró realizar un movimiento preciso.

Aunque el error articular presentado en la figura 4.10 no muestra errores demasiado grandes, estas variaciones y la falta de precisión en los motores no logran realizar el movimiento adecuado. El máximo error se presenta en la articulación 1 y 3 siendo este de 5 grados.

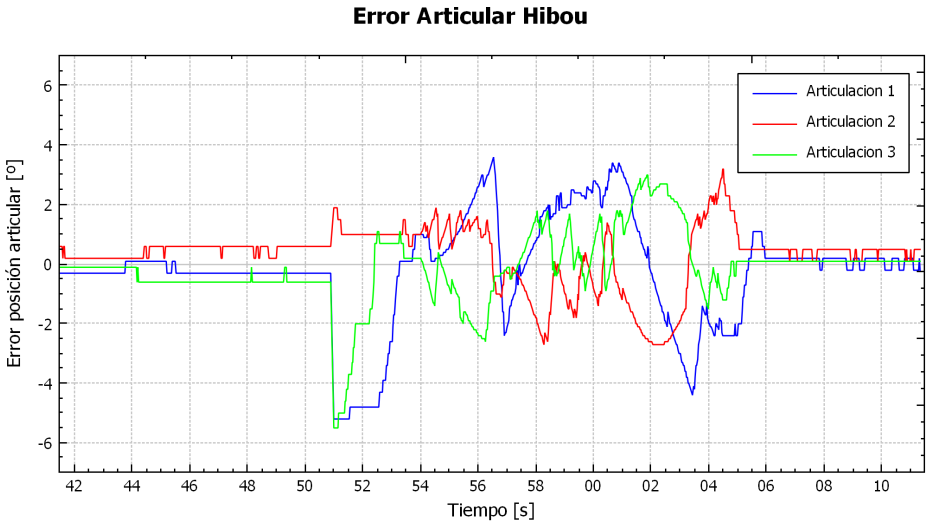

Figura 4.10: Error articular - Hibou.
Fuente: Elaboración propia.

Finalmente el error cartesiano en la figura 4.11 muestra un error máximo de 4,8 cm para este tipo de consigna.

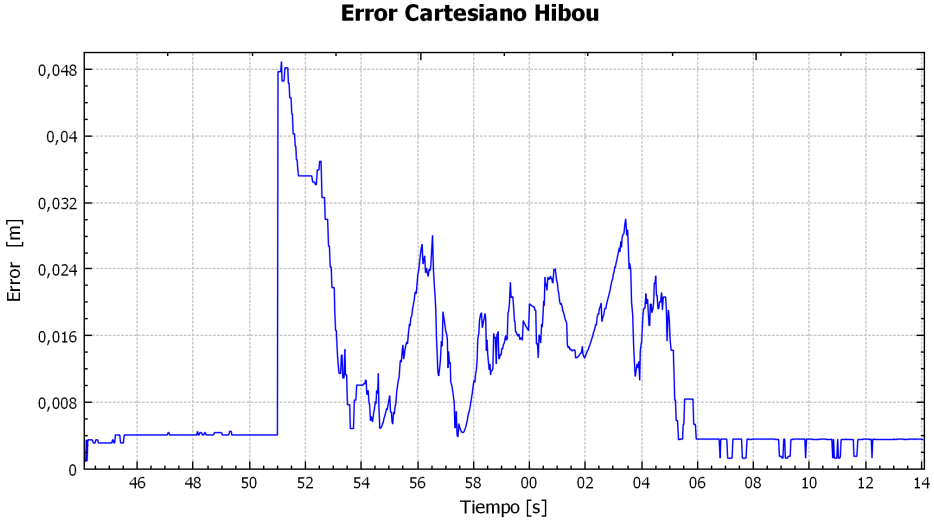

Figura 4.11: Error cartesiano - Hibou.
Fuente: Elaboración propia.

4.5.2. Lapbot

La consigna realizada tiene centro en $x = 30$, $y = 5$ cm y un radio equivalente a 2 cm. El punto de inicio en z está ubicado a los 0 cm y finaliza en 4 cm (ver figura 4.12).

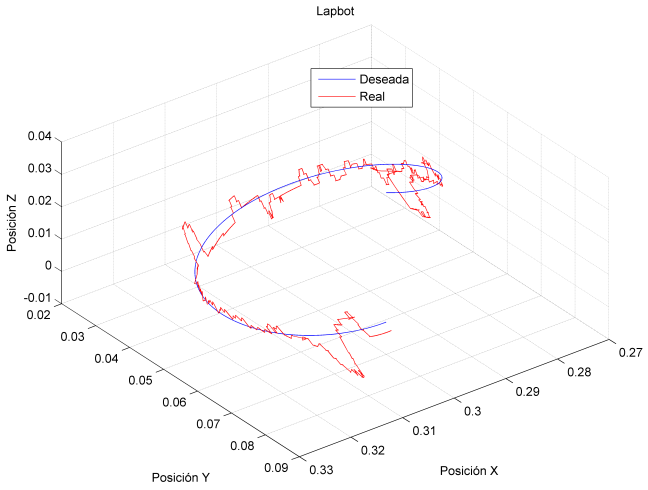

Figura 4.12: Consigna 3D realizada por el robot real Lapbot.
Fuente: Elaboración propia.

Inicialmente el tiempo de realización de la consigna fue 10 segundos, los resultados se observan en la figura 4.13, donde además se observa que la articulación 1 (prismática) es bastante lenta y no puede seguir rápidamente las referencias entregadas.

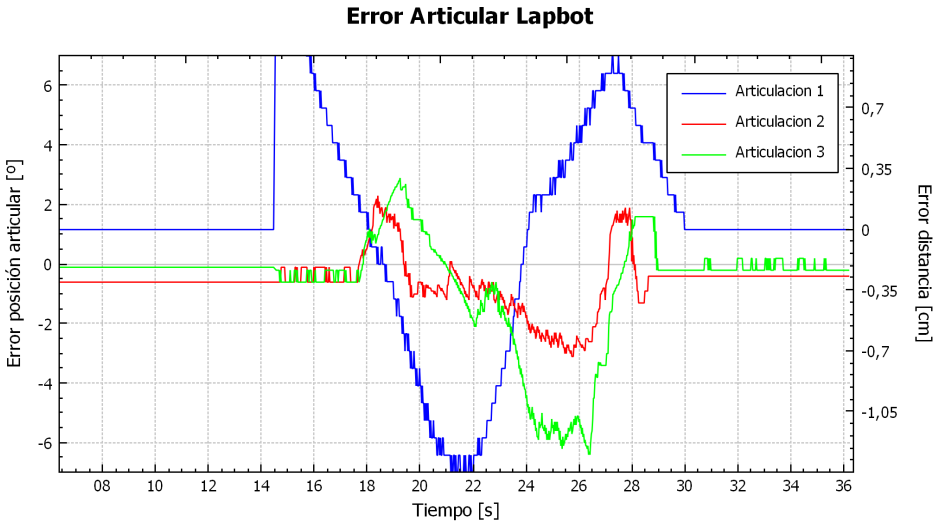

Figura 4.13: Error articular - Lapbot (ts = 10s).
Fuente: Elaboración propia.

Por tanto se modificó y amplió el tiempo de realización de la consigna siendo este 20 segundos. Los otros parámetros no fueron alterados.

El error articular presentado en la figura 4.14 muestra un error máximo de 3º en la articulación 2 por un pequeño instante de tiempo, sin embargo para la articulación 3 el error no supera los 2º. Para la articulación 1 el error no supera los 0,35 cm, siendo el motor lineal el más preciso de todos los actuadores. Finalmente el error cartesiano en la figura 4.15 muestra un error máximo de 1,9 cm.

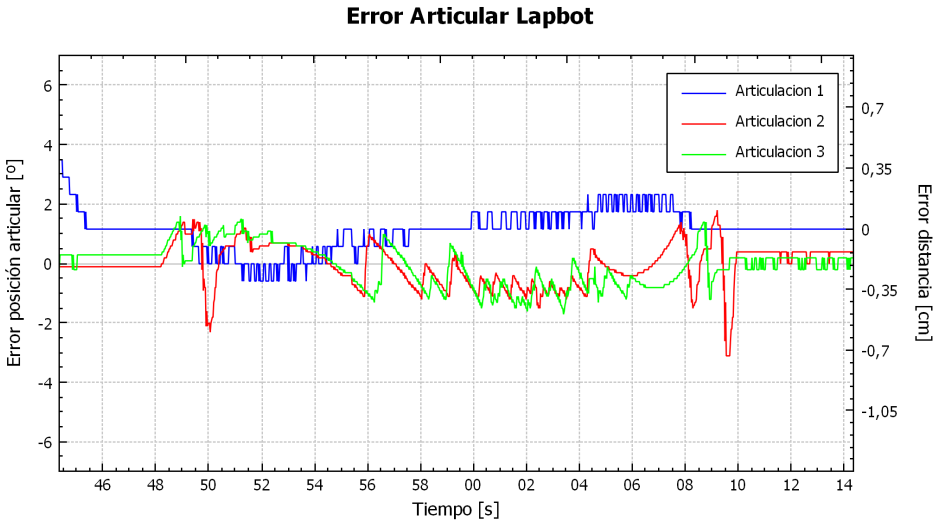

Figura 4.14: Error articular - Lapbot (ts = 20s).
Fuente: Elaboración propia.

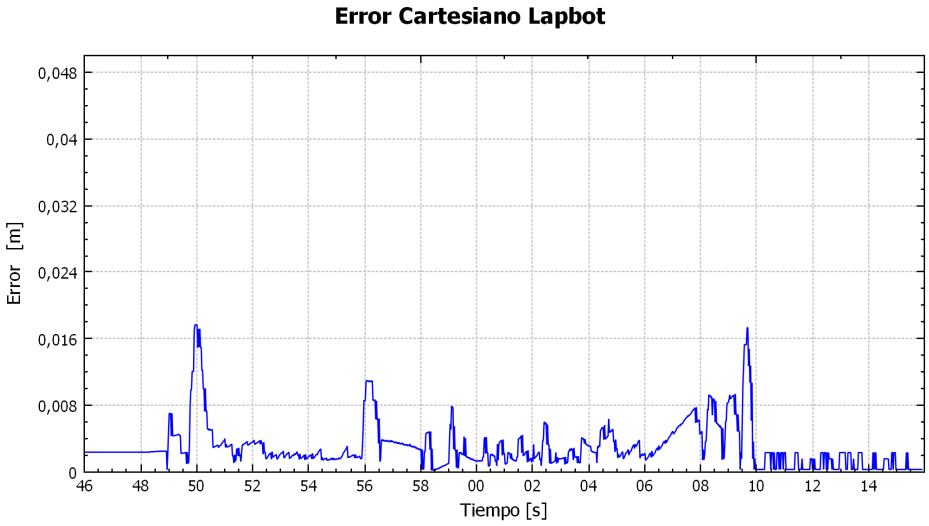

Figura 4.15: Error cartesiano - Lapbot (ts = 20s).
Fuente: Elaboración propia.

A pesar de contar con los mismos motores, las características de diseño de Lapbot hacen que este solo tenga lidiar con las fuerzas ejercidas por los motores, las 2 últimas articulaciones tienen movimientos horizontales, es decir que se desplazan sobre el eje y de tal manera que estas no tienen que generar movimientos fuertes que exijan alto torque, además el motor de la primera articulación realiza movimientos altamente precisos, por otro lado la estructura y diseño de Hibou hacen que este aparte de las fuerzas ejercidas por los motores, también deba lidiar con las fuerzas generadas por la gravedad en función de su masa, ya que dos de sus primeras tres articulaciones se mueven en forma ascendente y descendente, siendo afectadas directamente por la gravedad. Por esta razón se demuestra porque el robot Lapbot tiene mayor precisión que Hibou.

Capítulo 5

Aspectos a considerar a lo largo del proyecto

La culminación de este trabajo de grado deja como resultado el control de los robots Hibou y Lapbot, que en conjunto con los trabajos de grado previos a este comprenden el primer prototipo de un sistema robótico para asistencia laparoscópica en la Universidad del Cauca.

En todo trabajo o proyecto tecnológico de gran escala, es esencial realizar prototipos iniciales en los cuales siempre se van a encontrar falencias y limitaciones respecto a la versión comercial.

Este capítulo expone puntos de vista y consideraciones, que debieron ser tomadas en los trabajos de grado realizados anteriormente (modelado, simulación y construcción) y que hubiesen aportado mejoras de forma significativa al desempeño de los robots.

5.1. Fase de modelado y simulación

El software de simulación en C++ se ejecuta y realiza los procesos de renderizado, manejo serial, solución numérica de ecuaciones, y muchos otros procesos de forma secuencial, esperando la culminación de cada proceso para empezar el próximo, de tal forma que el tiempo empleado para realizar y completar todos los procesos para cada ciclo toma un tiempo muy alto, forzando a incrementar el tiempo de muestreo a un valor muy elevado respecto al simulado en Matlab.

Para el caso de la simulación del controlador en Matlab, se considera que no se tuvieron ciertos aspectos reales pensando en una implementación, y por tanto la caracterización de los actuadores no fue la pensada, donde solo se tomó en cuenta un solo parámetro: torque.

Un análisis más detallado pudo haber sido realizado teniendo en cuenta ciertas consideraciones:

- Una limitación de torque para la salida del controlador previa una investigación de motores y características disponibles en el mercado, esto sin restarle importancia a que debe obtenerse un error mínimo.
- Siguiendo estas especificaciones, un análisis de las velocidades resultantes de las articulaciones entregadas por el MDD, que aunque constituyen un parámetro aproximado, hubiesen entregado un nuevo parámetro característico para la adquisición de motores.
- Los tiempos de simulación se realizaron en tiempos poco convenientes, si bien representarían las expectativas para una versión final, no son viables para un primer prototipo. Estos parámetros de torque, velocidad y precisión solo existen en simulación y no representan parámetros físicos de motores existentes comercialmente, o por el contrario, demasiado costosos.

Se concluye que una correcta investigación interpolando los resultados obtenidos de parámetros como torque, velocidad y error, permitirían una mejor caracterización de los actuadores para cualquier robot.

5.2. Fase de diseño CAD y construcción

Cuando se quiere pasar de un diseño CAD a la construcción real hasta el más mínimo detalle cuenta, incluso usando maquinas CNC para realizar el corte de piezas, si los materiales elegidos no son los adecuados ya sea por costos o facilidad de manipulación, estos pueden causar problemas y generar inconvenientes mecánicos.

Los trabajos de grado asociados a la construcción de los robots tenían como finalidad la verificación de la movilidad de los mismos, la cual fue lograda satisfactoriamente. Sin embargo la falta de experiencia y complejidad del trabajo dejaron falencias mecánicas en los acoples de las articulaciones (motores) con las uniones de las piezas.

Los robots fueron construidos en triplex, un tipo de madera de bajo costo y de fácil uso, sin embargo añade mucho peso a la estructura y su deterioro en el tiempo es fuertemente visible; se sugiere el uso de materiales que pueden reemplazar el triplex y alivianar la estructura mejorando la movilidad y brindando una mejor estética con costos no tan elevados, estos son: acrílico, termoplásticos y fotopolímeros.

Los motores juegan un papel fundamental en la movilidad de los robots, los parámetros para su elección fueron torque de salida (criterio: simulación) y velocidad (criterio: debe generarse un movimiento lento, pero sin tener una referencia), dejando como resultado la elección de motorreductores de alto torque y una velocidad nominal de 75RPM, pero se dejó de lado un parámetro muy importante como lo es la *precisión* que estos pueden proporcionar.

La velocidad nominal de los motorreductores elegidos es incluso alta comparada con motorreductores de características de par semejante que pueden encontrarse en el mercado. A todo esto se le suma finalmente los juegos mecánicos asociados a los engranes de la caja reductora los cuales son muy altos. Estos juegos se reflejan en el extremo final de la articulación de acuerdo al ángulo de juego de los engranes y la distancia entre articulaciones.

En la construcción de los robots se instalaron motorreductores de características semejantes en cada una de las articulaciones rotoides, los cuales fueron elegidos de acuerdo al máximo toque necesario para mover una articulación determinada de acuerdo a los resultados de la simulación, instalando así dos motorreductores sobre dimensionados en las articulaciones rotoides restantes, generando mayor peso a la estructura.

El motor de desplazamiento lineal posee una velocidad muy lenta, si bien el torque y la precisión están sobredimensionadas, funcionan para el desarrollo del proyecto, sin embargo el voltaje de polarización de este elemento es a 24VDC en comparación con los otros motores rotativos, que funcionan a 12VDC impidiendo la estandarización de las fuentes de voltaje. Por otro lado para la medición de la posición del vástago del motor de desplazamiento lineal se utiliza un potenciómetro deslizante siendo este un elemento adicional, sin embargo el fabricante ofrece en su catálogo motores de similares características al elegido pero incluye en su estructura un sensor óptico para medición de la posición y su funcionamiento a 12 VDC.

Los sensores implementados en cada articulación rotoide son potenciómetros de alta precisión, sin embargo estos tiene un recorrido de 3 vueltas para todo su rango de trabajo. Los robots realizan recorridos no mayores a 50 grados en funcionamiento normal, e incluso en casos de errores o caídas accidentales de las articulaciones, estos no superarían los 270 grados de movilidad. Las 3 vueltas del potenciómetro son innecesarias causando que el rango del potenciómetro sea desperdiciado. De igual forma se pueden encontrar motorreductores que incluyen un *encoder*, ayudando a tener una medición más exacta evitando la adaptación de un elemento externo a la estructura.

La fuentes de voltaje aunque en los datos de la placa de fabrica cumplen las especi-

ficaciones técnicas necesarias para este proyecto, el uso durante distintas pruebas en el laboratorio muestra claramente que son inestables en su salida de voltaje ante exigencias de corrientes altas. Para este caso la corriente requerida al arranque y continuo uso de los motores es elevada y puede generar un calentamiento en las fuentes, dado que estas son para uso en computadores y no están diseñadas para soportar altas corrientes por largos periodos de tiempo. Sumado a esto al exigir una corriente alta, los voltajes de salida se reducen significativamente y generando oscilaciones, impidiendo que estas puedan usarse en aplicaciones que exijan alta precision.

5.3. Fase de control

La implementación de un controlador en simulación, aparte de verificar la controlabilidad del sistema, debe brindar una aproximación de las constantes del controlador PID, sin embargo en la dinámica de los modelos de los robots no se consideró la dinámica de los motores, debido a no contar con sus parámetros dinámicos, dado que no se contaba con una hoja de datos detallada.

Sin los parámetros dinámicos de los motores, la creación y adición de un modelo de motor DC a los robots era bastante complicada, y además no aportaba mucho realizar la aproximación a un modelo general de parámetros ideales. Por otro lado la estimación del modelo del motor utilizando técnicas de identificación se alejaba de los objetivos del proyecto.

Por esta razón, las constantes del controlador PID implementado en Arduino se sintonizaron partiendo desde el comienzo utilizando la misma técnica de sintonización que se usó para el controlador PID inicialmente, pero sin tener en cuenta los parámetros hallados en simulación.

La tarjeta Arduino implementada para este proyecto posee un ADC de 10 bits de resolución. Con las exigencias de precisión y exactitud que requiere un sistema robótico los canales de ADC para hacer lectura de los sensores deber ser de 16 bits, y además el Arduino no es muy exacto en las mediciones realizadas, presentando variaciones en cada lectura. Sin embargo este dispositivo se utiliza en este proyecto en primera instancia por tenerlo como recurso académico en la Universidad del Cauca, y por otro lado los juegos mecánicos presentes en los robots evitan obtener una precisión muy alta, por tanto usar un dispositivo de mayor resolución en la lectura no aportaría una mejora notoria, solo aumentaría los costos sin llegar a utilizar su verdadera capacidad.

En general las dos estructuras de los robots quedaron con peso excesivo, para Lap-

bot la articulación prismática realiza movimientos muy lentos respecto al motorreductor perjudicando el control, y para Hibou las articulaciones rotoides dos y tres realizan movimientos ascendentes y descendentes, siendo estas afectadas positiva y negativamente por la gravedad y su peso.

La estructura de Hibou en cuanto a movilidad quedo funcional, pero el peso es excesivo y la segunda articulación no es capaz de mover la carga asociada cuando ésta debe moverse en contra de la gravedad, exigiendo esfuerzos de control elevados que en ocasiones no son suficientes o generan movimientos bruscos perturbando así el lazo de control en el robot, incluso cuando el robot esta en reposo para ciertas posiciones, el motor no es capaz de soportar el peso ejercido por la estructura rompiendo la inercia del motor, generando movimientos bruscos en favor de la gravedad.

En las articulaciones ya sean activas o pasivas se debe hacer medición para obtener el error cartesiano, sin embargo en ambos robots se cuenta únicamente con sensores en las primeras tres articulaciones (activas), lo cual nos lleva a utilizar las posiciones ideales de simulación y mezclarlas con las mediciones reales para estimar la posición cartesiana del órgano terminal de los robots, derivando en resultados no propios de los movimientos y posiciones de los robots.

5.4. Problemas presentados y recomendaciones

A lo largo de proyecto se presentaron diversas anomalías propias de trabajos de implementación, que muchas veces pueden generar demoras en los proyectos por problemas que son aparentemente insignificantes.

5.4.1. Protocolo de comunicación

Uno de los mayores problemas fue realizar una correcta transmisión de datos con una la velocidad que garantizara el tiempo de muestro utilizado en simulación (1ms). Se puede encontrar fácilmente información de cómo realizar transmisión y recepción por serial, pero la dificultad radica cuando se deben usar periodos de transmisión muy cortos, tal como es el caso en este proyecto.

Realizar la correcta lectura no garantiza que se van a presentar errores, debido a la alta velocidad estos se pueden presentar sin una causa justificada, por tanto bajo estos términos se recomienda usar detectores de errores o caracteres de inicio y fin de trama al usar velocidades altas (entre más alta es la velocidad, más susceptible es a perturbaciones).

En primer paso se opta por usar la máxima velocidad de transmisión, pero esto conlleva un inconveniente: transmisor y receptor deben ser capaces de realizar su proceso en un periodo corto de tiempo, si uno de los dispositivos no es capaz, o no se encuentra disponible para enviar o leer los datos rápidamente, el *Buffer*[1] se desbordará.

Para el caso del dispositivo Arduino, el *Buffer* sobrepasará su capacidad de almacenamiento rápidamente (aproximadamente 10 tramas recibidas para una cadena 12 datos) y provocará que este colapse, reiniciándose automáticamente después de cierto periodo de tiempo (aproximadamente 10 a 15 segundos). Para el caso del computador, el *Buffer* es mucho más grande, provocando que los datos queden almacenados hasta que sean leídos, causando una falsa lectura de datos en el programa C++, el cual mostraría datos aparentemente erróneos, pero en realidad se leen datos antiguos almacenados en el *Buffer*. Bajo pruebas se comprobó que el computador es capaz de almacenar datos hasta por un lapso de 10 segundos para un periodo de transmisión de 10 ms.

Por otro lado a lo largo del proyecto se optó por usar un segundo dispositivo serial[2], que constituye una recomendación importante a la hora de trabajar con transmisión serial. Este dispositivo permitió comprobar muchas variables de interés presentes en el Arduino, dado que el puerto serial principal se encuentra ocupado en la transmisión de datos con C++ y no puede usarse para otros propósitos.

Sin que se hayan realizado pruebas, pero teniendo el conocimiento de la existencia de protocolos de comunicación más rápidos y mejor estructurados, se recomienda usar protocolos de comunicación como USB y Ethernet para próximos proyectos.

5.4.2. Sensores

Cuando se requieren mediciones precisas y a alta velocidad se deben tener en cuenta los tiempos que tarda cualquier dispositivo en medir y cuantificar una señal análoga en digital, además si se debe trabajar con sensores externos, se debe tener en cuenta que no todos poseen el mismo comportamiento y al momento de adaptarlos al sitio de medición es posible que todos presenten posiciones diferentes.

El primer paso es realizar una caracterización del sensor con el fin de verificar su comportamiento usando la polarización que se va a utilizar en el proyecto, para varios sensores

[1] *Buffer*: Variable que almacena temporalmente los datos provenientes o salientes en un dispositivo serial

[2] El CP2102 es un dispositivo para uso en sistemas embebidos que requieren una comunicación USB hacia un ordenador tendiendo como punto de partida un puerto UART o Serial. Este se puede simplemente conectar a un bus USB de una computadora, o haciendo uso de un cable de extensión con un conector hembra estándar de tipo A y aparecerá como un puerto COM estándar.

hacer las mismas pruebas de caracterización para no obtener resultados que generen confusión, y por ultimo generar una ecuación que describa el comportamiento de los sensores para que esta pueda ser implementada en código en el sistema de adquisición de datos y ser modificada según los requerimientos.

Cuando se vayan a instalar o adaptar los sensores se recomienda utilizar la misma posición para todos con el fin de estandarizarla, evitando confusiones ante daños que requieran cambios de sensor o movimientos indeseados del sensor.

Se recomienda usar una fuente de alimentación independiente a la que se utilice para los actuadores, o regular la tensión con integrados. Esto permite eliminar una gran cantidad de ruido, mantener una lectura estable cuando los actuadores se activen y puedan generar una caída de voltaje por exigencias de corriente.

Todos los dispositivos que utilizan ADC realizan una lectura del mundo real y la transforman en una señal digital, esto ya constituye una aproximación por tanto, los resultados siempre serán variantes y rara vez iguales. Para reducir el ruido en las mediciones se usan filtros físicos o filtros por software como son por ejemplo el promedio y la mediana para minimizar valores fuera de rango.

Para la adquisición de sensores, se recomienda que estos sean incluidos en el motor para evitar problemas en el montaje y adaptación, y en caso de ser un sensor digital tener en cuenta la precisión en la medida, la velocidad de variación y transmisión de la señal.

5.4.3. Actuadores

Para este caso solo se hace referencia a los motorreductores que son los utilizados en el proyecto.

Una correcta caracterización de la corriente nominal y de arranque conduce a una buena selección de la fuente de alimentación. Se debe tener en cuenta el voltaje y corriente mínimo a partir del cual el motor inicia su movimiento, permitiendo acotar los extremos del controlador y evitando que ingrese en zonas donde no ejerza ningún cambio o esfuerzo de control.

Es de tener en cuenta que la caracterización genera parámetros que no serán iguales para todos los motorreductores, dado que dependen del torque que deben realizar, sentido de giro, polarización, puente H entre otros.

Se recomienda para este tipo de proyectos, en caso de usar motorreductores, que estos trabajen a velocidades menores a 30 RPM, que el juego de los engranes sea muy bajo y la reducción de las cajas sea mayor a 500 veces.

5.4.4. Fuentes de voltaje

Una buena fuente de alimentación permite brindar un buen funcionamiento tanto para sensores como actuadores. Esta brindará una buena estabilización de voltaje para los sensores, impidiendo variaciones, ruido o sobrecalentamiento. Así mismo los actuadores podrían llegar a funcionar a su máxima capacidad en caso de que la aplicación lo requiera, sin que la fuente sufra variaciones de voltaje o sobrecalentamiento.

De igual forma en caso de no tener una fuente capaz de soportar toda la carga del proyecto, se pueden utilizar dos fuentes para alimentación, una para realizar control y la otra para manejo de potencia.

5.4.5. Documentación

Se recomienda que para trabajos de implementación donde se planean proyectos futuros, dejar una amplia documentación del trabajo realizado y aún más detallado los aspectos técnicos que sean importantes, es decir hojas de datos, elementos utilizados, modificaciones realizadas, pruebas, etc. Si bien algunos aspectos pueden ser considerados no relevantes y pueden ser realizados bajo pequeñas pruebas, representaran un gasto de tiempo y esfuerzo. Una documentación detallada del trabajo, resultados, problemas y técnicas implementadas permiten que futuros lectores apropien el trabajo realizado en nuevos proyecto sin demoras o problemas de comprensión.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

El dispositivo Arduino es una buena plataforma embebida para la implementación de controladores en aplicaciones que no requieran mucha precisión, presentando un lenguaje de programación de fácil uso y entendimiento con amplia y detallada información en su plataforma virtual, además presenta múltiples ventajas en cuanto a módulos de expansión, accesorios, estabilidad en funcionamiento y bajo costo. Como desventaja al igual que múltiples microcontroladores, su forma de ejecución del código es secuencial manejando un solo hilo a la vez, entonces si se tienen múltiples lecturas ADC se incrementa el tiempo de procesamiento y se limitaría a aplicaciones no que requieran altas velocidades.

La versatilidad del entorno de programación de Arduino permite que el control PID implementado sobre pueda ser utilizado en cualquier robot independiente del número de articulaciones, la forma en que se codificó el controlador permite que cada articulación requiera solo dos líneas de código para ser añadido al programa.

Además de forma general el controlador PID sigue siendo una alternativa funcional para el control de robots tipo serie donde no se tenga conocimiento del modelo matemático. Los controladores implementados presentaron buen desempeño en el seguimiento de posiciones para las tres articulaciones medidas aun con las limitaciones mecánicas presentadas

Mostrando así que el primer prototipo de los robots Hibou y Lapbot cumple el objetivo de mostrar su capacidad de realizar movimientos en un entorno quirúrgico respetando el paso por el trocar, sin embargo debido a las diferentes dificultades mecánicas que se presentaron, los robots no pueden ser tomados como una versión que logre mostrar el potencial de sus modelos y diseños.

El calculo del error cartesiano se realizó con datos reales y simulados, lo que permitió

encontrar una aproximación de la posición del órgano terminal validando así el funcionamiento de controlador PID.

Por otro lado, el software base del cual surgió RELAS fue desarrollado para ejecución por eventos, estos se ejecutan de forma secuencial según el orden de atención, desaprovechando las nuevas características de los nuevos procesadores. En este proyecto se agregaron funciones que se ejecutaron en paralelo al evento normal de ejecución, mejorando el tiempo de ejecución del software, el cual presenta una mejora significativa que se puede implementar en nuevos desarrollos que requieran robustez.

Por ejemplo, usar programación orientada a objetos que ofrece un mejor desempeño en programas que requieran un alto rendimiento. Con herramientas software como *Qt* o *QCustomPlot* de código abierto basadas en c++ se puede acceder a una amplia documentación, ayudas y tutoriales, que permiten utilizar al máximo las capacidades de un software sin generar costos en licencias o soporte en su implementación.

6.2. Trabajos futuros

Mejorar la estructura mecánica de los robots de acuerdo a las apreciaciones dadas en las secciones anteriores, añadiendo los sensores y motores en las articulaciones que hacen falta, además implementar las articulaciones que no fueron construidas y que son necesarias para el correcto funcionamiento de los robots en un entorno quirúrgico.

Estas mejoras permitirían implementar los instrumentos reales (tijeras, endoscopio, aguja insuflación, grapadora) en el robot que corresponda, con el fin de aproximar los robots a una versión comercial, probando y verificando el correcto funcionamiento de instrumentos.

Además se puede adicionar una interfaz háptica para la realimentación de fuerzas ya sea de forma visible o sensorial, que aumente el realismo durante pruebas reales, haciendo que la herramienta software sea capaz de mostrar la respuesta real de los robots en el entorno quirúrgico.

Desarrollar tambien un sistema de control utilizando un procesador más robusto (BeagleBone, Raspberry pi u otro) con mejores periféricos para lectura de los sensores, que permita la incorporación de una interfaz de usuario para manipulación directa sin uso del computador, haciendo de los robots un sistema móvil y de fácil transporte para ser exhibido y utilizado.

Esto acompañado de un nuevo diseño o mejora del actual software para la manipulación de los robots, agregando estructura por hilos, mejorando el rendimiento del programa

(velocidad de cálculo, capacidad de respuesta), con llevando a implementar un controlador con tiempos de muestreo menores a 10 milisegundos.

Y finalmente realizar la identificación de los modelos matemáticos de los robots con el fin de implementar técnicas de control más robustas.

Bibliografía

[1] Cirugía Mínima Invasiva y Obesidad, "Cirugía mínima invasiva." [Online], Disponible en: http://www.cmiyo.com/index.php?id_categoria=11&id=9. (Consultado en Febrero de 2015).

[2] R. Castillo, R. Pérez, J. García, R. Álvarez , "La importancia de la cirugía laparoscópica para el cirujano general," *Revista Salud en Tabasco*, vol. 12, no. 2, pp. 443 – 448, 2006.

[3] J. Albani, "The role of robotics in surgery: A review," in *Science & Technology Review*, vol. 104, 2007.

[4] O. Vivas, "Aplicaciones de la robótica al campo de la medicina," *Revisa Pulsos*, vol. 9, pp. 32 – 38, 2007.

[5] P. Ricci, R. Lema, V. Solá, J. Pardo, E. Guiloff , "Aplicaciones de la robótica al campo de la medicina desarrollo de la cirugía laparoscópica: pasado, presente y futuro. desde hipócrates hasta la introducción de la robótica en laparoscopia ginecológica," *Revista chilena de obstetricia y ginecología*, vol. 73, no. 1, pp. 63 – 75, 2008.

[6] Forbes, "Intuitive surgical expects q4 growth on rising da vinci xi sales." [Online], Disponible en: http://www.forbes.com/sites/greatspeculations/2015/01/15/intuitive-surgical-expects-q4-growth-on-rising-da-vinci-xi-sales/. (Consultado en Enero de 2016).

[7] O. Castillo, R. Sanchez, "Bases laparoscópicas de la cirugía robótica," *Archivos Españoles de Urología*, vol. 60, no. 4, pp. 357 – 362, 2007.

[8] All About Robotic Surgery, "Surgical robots - a brief history." [Online], Disponible en: http://allaboutroboticsurgery.com/surgicalrobots.html. (Consultado en Febrero de 2016).

[9] A. Lanfranco, A. Castellanos, J. Desai, W. Meyers, "Robotic surgery: A current perspective," *Annals of Surgery*, vol. 239, no. 1, pp. 14 – 16, 2004.

[10] M. Oddsdóttir, G. Birgisson, "Aesop: A voice-controller camera holder," in *Primer of Robotic and Telerobotic Surgery* (G. Ballantyne, Jacques Marescaux, P. Giulianotti, ed.), Introduction to surgical Robots and Telerobots, p. 35, 2004.

[11] A. Córdova, G. Ballantyne, "Sistemas quirúrgicos robóticos y telerobóticos para cirugía abdominal," *Revista de Gastroenterología*, vol. 23, no. 1, pp. 58 – 66, 2003.

[12] S. Shew, D. Ostlie, G. Holdcomb III, "Robotic telescopic assistance in pediatric laparoscopic surgery," *Pediatric Endosurgery & Innivative Techniques*, vol. 7, no. 4, pp. 372 – 376, 2003.

[13] J. Marescaux, J. Leroy, M. Gagner, F. Rubino, D. Mutter, M. Vix, S. Butner, M. Smith, "Transatlantic robot-assisted telesurgery," *Nature*, vol. 413, pp. 379 – 380, 2001.

[14] Intuitive Surgical, "Da vinci surgery - minimally invasive robotic surgery with the davinci surgical system." [Online], Disponible en: http://www.davincisurgery.com. (Consultado en Febrero de 2015).

[15] C. Amate, "Robot quirúrgico da vinci: máxima precisión en el quirófano." [Online], Disponible en: http://blogthinkbig.com/robot-quirurgico-da-vinci/. (Consultado en Febrero de 2015).

[16] SOFAR, "The new age of minimally invasive surgery has just began." [Online], Disponible en: http://www.alf-x.com/. (Consultado en Noviembre de 2015).

[17] Eindhoven University of Technology, "Sofie." [Online], Disponible en: https://www.tue.nl/en/research/research-institutes/robotics-research/projects/sofie/. (Consultado en Noviembre de 2015).

[18] German Aerospace Center, "Mirosurge - telemanipulation in minimally invasive surgery." [Online], Disponible en: http://www.dlr.de. (Consultado en Noviembre de 2015).

[19] J. Berná, F. Marciá, V. Gilart, H. Ramos, A. Ferrándiz, "Sistema de control para robots inspirado en el funcionamiento y organización de los sistemas neurorreguladores humanos," in *Desarrollo de grandes aplicaciones de red: V Jornadas*, pp. 59 – 75, JDARE, Alicante, España 2008.

[20] E. Muñoz, V. Mosquera, C. Gaviria, O. Vivas, "Evaluación del desempeño de diversos controladores avanzados aplicados a un robot manipulador tipo scarañ̃," *Revista Ingeniería y Desarrollo*, vol. 29, no. 2, pp. 202 – 223, 2011.

[21] R. Carmona, "Análisis de estabilidad en controladores pid y neuronales pid sobre robots," Tesis de Doctorado, Departamento de Control Automático, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México 2013.

[22] O. Cieza, "Implementacion de una estacion de trabajo mediante un robot serial de 3 grados de libertad," Tesis de Pregrado, Ingeníeria Electrónica, Universidad Pontifica del Perú, Perú 2011.

[23] G. Selim, N. El-Amary, D. Aboul, "Force signal tuning for a surgical robotic arm using pid controller," *International Journal of Computer Theory and Engineering*, vol. 4, no. 2, pp. 58 – 66, 2012.

[24] J. Orrante-sakanassi, V. Santibañez, R. Campa, "On saturated pid controllers for industrial robots the pa10 robot arm as case of study," *Advanced Strategies for Robot Manipulators*, vol. ISBN: 978-953-307-099-5, no. 2, pp. 217 – 248, 2010.

[25] P. Ouyanga, V. Panoa , T. Dama, "Pid position domain control for contour tracking," *International Journal of Systems Science*, vol. 46, no. 1, pp. 111 – 124, 2013.

[26] D. Pratama, E. Henfri, F. Ardilla, "Movement control of two wheels balancing robot using cascaded pid controller," *International Electronics Symposium (IES)*, pp. 100 – 105, 2015.

[27] F. Piltan, A. Badri, J. Meigolinedjad, M. Keshavarz, "Adaptive artificial intelligence based model base controller: Applied to surgical endoscopy telemanipulator," *I.J. Intelligent Systems and Applications*, vol. 9, pp. 103 – 115, 2013.

[28] J. Iqbal, M. Imran, A. Khan, M. Irfan, "Towards sophisticated control of robotic manipulators: An experimental study on a pseudo-industrial arm," *Strojniikki vestnik - Journal of Mechanical Engineering*, vol. 61, no. 7 - 8, pp. 465 – 470, 2015.

[29] P. Begoa, A. Zubizarreta , I. Cabanes , "Control basado en modelo para robots paralelos con sensorización redudante," *XXXVI Jornadas de Automática*, pp. 416 – 423, 2015.

[30] O. Vivas, "Control predictivo de un robot tipo scara," *Revista Chilena de Ingeniería*, vol. 14, no. 2, pp. 135 – 145, 2006.

[31] F. Lara, J. Rosário, D. Dumur, P. Wenger, "Application of predictive control techniques within parallel robot," *Revista Controle y Automacao*, vol. 23, no. 5, pp. 530 – 540, 2015.

[32] S. Salinas, "Modelado, simulación en 3d y control de un robot para cirugía laparoscópica," Tesis de Maestría en Electrónica y Telecomunicaciones, Universidad del Cauca, Colombia, 2009.

[33] S. Salinas, A. Vivas, "Lapbot: Robot para cirugía laparoscópica," Libro de Actas Simposio Cea de Bioingeniería, pp. 33-38, España, 2009.

[34] S. Salinas, A. Vivas, "Modelado, simulación y control del robot para cirugía laparoscópica lapbot," *Revista Chilena de Ingeniería*, vol. 17, no. 3, pp. 317 – 328, 2009.

[35] C. Méndez, V. Torres, "Diseño y simulación en 3d de un robot porta endoscopio para cirugía laparoscópica," Tesis de Pregrado, Ingeniería en Automática Industrial, Universidad del Cauca, Colombia, 2010.

[36] B. Garcés, O. Mora, O. Vivas, "Estudio del uso de robots industriales como asistentes en operaciones de laparoscopia," *Revista Facultad de Ingeniería Universidad de Antioquia*, no. 47, pp. 91 – 102, 2009.

[37] C. Fernández, H. Guástar, "Diseño de un robot tipo pa-10 virtual para aplicaciones quirúrgicas," Tesis de Pregrado, Ingeniería en Automática Industrial, Universidad del Cauca, Colombia, 2014.

[38] C. Monserrat, O. López, U. Meier, M. Juan, V. Grau, J. Gil, J. Lozano, M. Alcañiz, "Simulador quirúrgico virtual para el entrenamiento en cirugías mínimamente invasivas," IX Congreso Nacional de Informática Médica, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, España, 2012.

[39] LAP Mentor, "Only simbionix provides basic and advanced surgical simulation training." [Online], Disponible en: http://simbionix.com/simulators/lap-mentor/. (Consultado en Noviembre de 2015).

[40] CAE Healthcare, "Cae healthcare." [Online], Disponible en: http://www.caehealthcare.com/. (Consultado en Noviembre de 2015).

[41] C. Muñoz, E. Lasso, C. Tosse, F. Ordóñez, A. Vivas, "Sistema virtual para el posicionamiento del robot porta endoscopio hibou a través de un casco," *Revista Universitaria en Telecomunicaciones, Informática y Control*, vol. 1, no. 1, pp. 59 – 65, 2012.

[42] D. Guzman, A. Vivas, "Software para la práctica de la robótica quirúrgica," *Ingeniería y Universidad*, vol. 19, no. 1, pp. 7 – 26, 2015.

[43] J. Samboní, C. Fuentes, "Base para sistema de entrenamiento quirúrgico: Robot hibou," Tesis de Pregrado, Ingeniería en Automática Industrial, Universidad del Cauca, Colombia, 2013.

[44] M. Chaparro, A. Caicedo, "Base para sistema de entrenamiento quirúrgico: robot lapbot," Tesis de Pregrado, Ingeniería en Electrónica y Telecomunicaciones, Universidad del Cauca, Colombia, 2013.

[45] Vishay, "Multi turn wirewound potentiometer model 533." [Online], Disponible en: http://www.vishay.com/docs/57065/533534.pdf. (Consultado en Febrero de 2015).

[46] Linak, "Linak, we improve your life." [Online], Disponible en: http://www.linak-latinamerica.com/productos/linear-actuators.aspx. (Consultado en Febrero de 2015).

[47] Microchip, "Pic18f4550." [Online], Disponible en: http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC18F4550. (Consultado en Noviembre de 2015).

[48] Pololu, "Pololu robotics & electronics." [Online], Disponible en: https://www.pololu.com/product/755. (Consultado en Marzo de 2015).

[49] Amazon, "High power dc motor drive module h-bridge." [Online], Disponible en: http://www.amazon.com/Zhangminivy-Module-H-bridge-Strong-Brakes/dp/B00BSULZCS. (Consultado en Febrero de 2016).

[50] Arduino, "Arduino - language reference." [Online], Disponible en: http://arduino.cc/en/Reference/HomePage. (Consultado en Marzo de 2015).

[51] O. Vivas, *Diseño y control de robots industriales: teoría y práctica*. Elaleph.com, 1ª ed., Buenos Aires, 2010.

[52] C. Rengifo, "Acción derivativa," Notas de clase - Electiva de PID, Ingeniería en Automática Industrial, Universidad del Cauca, Colombia, 2013.

[53] Himatix, "Debugging serial (rs232c) cables." [Online], Disponible en: http://www.himatix.com/resources/tech/serial.html. (Consultado en Diciembre de 2015).

[54] EducaChip, "Cómo y por qué usar las interrupciones en arduino." [Online], Disponible en: http://www.educachip.com/como-y-por-que-usar-las-interrupciones-en-arduino/. (Consultado en Diciembre de 2015).

[55] J. Boxall, "Tutorial: Arduino timing methods with millis()." [Online], Disponible en: http://tronixstuff.com/2011/06/22/tutorial-arduino-timing-methods-with-millis/. (Consultado en Diciembre de 2015).

[56] Precision Microdrives, "Pwm frequency for linear motion control." [Online], Disponible en: http://www.precisionmicrodrives.com/application-notes-technical-guides/application-bulletins/ab-022-pwm-frequency-for-linear-motion-control. (Consultado en Diciembre de 2015).

[57] J. Frost, "Why you need to check your residual plots for regression analysis." [Online], Disponible en: http://blog.minitab.com/blog/adventures-in-statistics/why-you-need-to-check-your-residual-plots-for-regression-analysis. (Consultado en Diciembre de 2015).

[58] J. Errington, "Experiments with an arduino - precise voltage measurement." [Online], Disponible en: http://www.skillbank.co.uk/arduino/measure.htm. (Consultado en Diciembre de 2015).

[59] J. Rodríguez, "Arduino - how to improve the realiability of sensors readings." [Online], Disponible en: http://blog.juanrc.com/2014/01/arduino-how-to-improve-realiability-of.html. (Consultado en Diciembre de 2015).

[60] L. Guerrero, C. Mendez, "Trabajo de clase de robótica," Maestría en Automática, Universidad del Cauca, Colombia, 2012.

[61] J. Meles, D. Battistelli, "Procesos de desarrollo de software y modelos de proceso (ciclos de vida)," Cátedra de Ingeniería de Software, Facultad Regional Córdoba, Universidad Tecnológica Nacional, Argentina, 2011.

[62] Ramon de Klein, "Serial library for c++." [Online], Disponible en: http://www.codeproject.com/Articles/992/Serial-library-for-C. (Consultado en Marzo de 2015).

[63] E. Eichhammer, "Qcustomplot." [Online], Disponible en: http://www.qcustomplot.com/. (Consultado en Diciembre de 2015).

[64] The Qt Company, "Qtconcurrent namespace." [Online], Disponible en: http://doc.qt.io/qt-4.8/qtconcurrent.html. (Consultado en Diciembre de 2015).

Anexos

Anexo A

Manual de usuario *RELAS*

A.1. Requisitos

El programa RELAS puede ser ejecutado en un equipo que cumpla con los requisitos *hardware* mínimos, sin embargo se puede observar que el programa tarda un largo tiempo en responder para realizar algunas tareas. Para obtener un mejor desempeño de ejecución del software se sugieren los requisitos *hardware* recomendados o superior, los cuales permitirán una mejor experiencia visual para el usuario.

Requisitos mínimos *hardware*

- Procesador doble núcleo 1,6 GHz.
- 2 GB de memoria RAM.
- 100 MB de espacio disponible en el disco duro.
- Tarjeta de vídeo compatible con DirectX 9.0 con una resolución de pantalla de 1024 x 768 o superior.

Requisitos recomendados *hardware*

- Procesador de cuatro núcleos 2,8 GHz o superior a 64-bit.
- 4 GB de memoria RAM.
- 100 MB de espacio disponible en el disco duro.
- Tarjeta de vídeo AMD Radeon HD 6450 o superior.

- Memoria gráfica: 512 MB

Requisitos del sistema

Se requiere como sistema operativo Windows 7, que incluya .NetFramework (2.0, 3.0, 3.5 y 4.0), Visual C++ redistributable (2005,2008,2010) y DirectX 9.0 para su funcionamiento.

Requisitos externos

- Joystick o gamepad con análogos.
- Nueva tarjeta electrónica.
- Teclado y mouse.

A.2. Instalación

Para instalar el software *RELAS* en su computadora, ejecute el instalador "RELAS Installer.exe" y siga las instrucciones. El software por defecto se instala en la carpeta "RELAS" en "Archivos de programa". La ubicación de la instalación puede ser modificada por el usuario.

A.3. Uso del programa

A.3.1. Ventanas del software

El software cuenta con dos ventanas, una de ellas es la ventana de avisos (ver figura A.1), en esta ventana se pueden apreciar todos los avisos de sucesos en el software, tales como errores en la comunicación serial o la conexión con éxito de la misma.

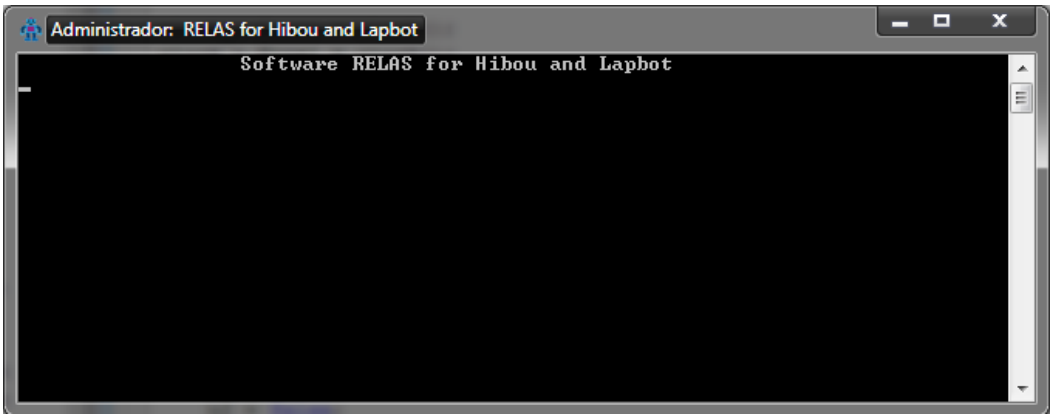

Figura A.1: Ventana de avisos

La segunda de las ventanas es la ventana principal (ver figura A.2), donde se visualiza el entorno tridimensional del espacio operacional donde se encuentran los robots y los menús para el despliegue de todas las funciones del software.

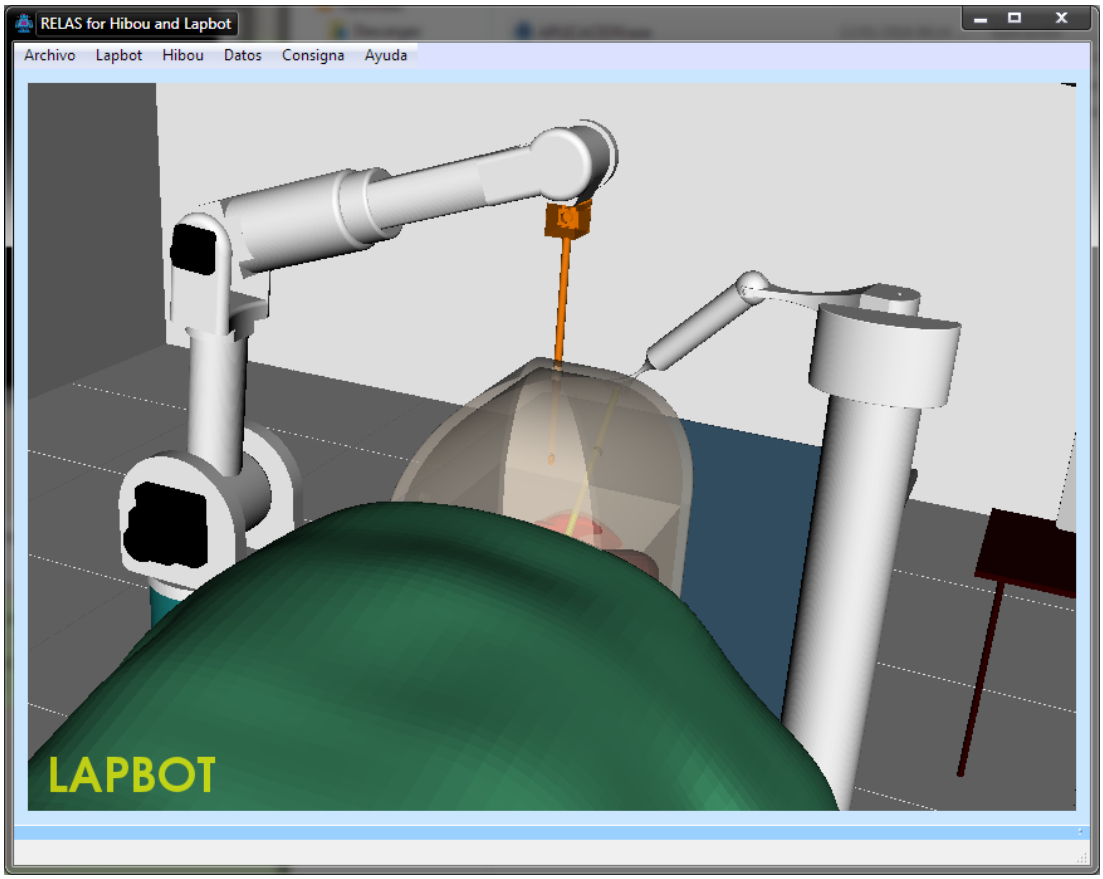

Figura A.2: Ventana principal para manipulación de los robots

En el entorno se visualiza una caja que simula la cavidad abdominal con una visión de

los órganos del cuerpo humano.

Para cambiar la posición de la cámara se arrastra el puntero en la dirección deseada y se usa la rueda de desplazamiento del *mouse* para hacer *zoom*.

En la parte inferior izquierda de la ventana se muestra el nombre del robot que se tiene seleccionado.

Nota: Antes de iniciar la aplicación se debe tener conectado el dispositivo *joystick*, de lo contrario se mostrará un aviso de error y la aplicación debe ser reiniciada.

Si se está utilizando un *gamepad* verificar que el *led* de los análogos este encendido antes de ejecutar la aplicación.

A.3.2. Comunicación serial

Una vez ejecutada la aplicación con el *joystick* conectado, se debe encender la fuente de alimentación de los sensores de los robots, después se deben mover los ejes del *joystick* o *gamepad* para verificar la movilidad de los robots, esto como una condición de seguridad. De no hacer el movimiento de los ejes del *joystick* no se podrá realizar la comunicación serial con la tarjeta electrónica.

Una vez realizadas las acciones previas y los robots puedan moverse gráficamente en el software se puede realizar la comunicación serial. Para ello nos ubicamos en la barra de menús se selecciona *Archivo - Comunicación Serial* donde se acceden a las dos opciones disponibles. Para comenzar la transmisión serial se selecciona *Iniciar* (ver figura A.3).

En la siguiente ventana se ingresa el número del puerto COM correspondiente al dispositivo Arduino, este número puede ser encontrado en el *Administrador Dispositivos*(una guía de cómo realizar este paso se puede encontrar en el siguiente enlace:

`http://windows.microsoft.com/es-co/windows/open-device-manager#1TC=windows-7`.

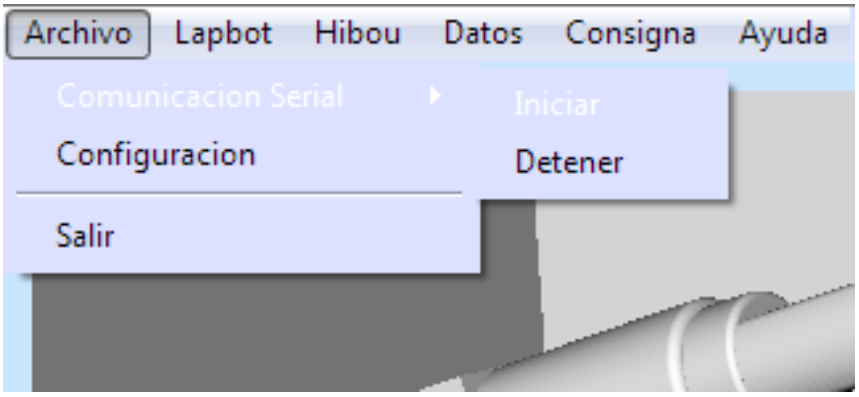

Figura A.3: Comunicación serial en c++

Si la conexión es exitosa, un mensaje lo indicara, de lo contrario se debe verificar que el dispositivo Arduino se encuentre correctamente conectado y el puerto COM corresponda al asignado. Diríjase a la sección A.4 *solución de problemas* para obtener ayuda si el puerto COM no inicia.

Para finalizar la comunicación de datos se selecciona el Menú *Archivo* - *Comunicación Serial* y dar click en *Detener*. Un mensaje mostrara que la conexión fue finalizada.

A.3.3. Joystick

Después de realizar la comunicación serial, se debe encender la fuente de alimentación para los componentes de potencia, la cual en este punto permitirá realizar movimientos con el *joystick*.

Los ejes cartesianos que describen la movilidad de los robots usando el joystick y la forma en cómo se puede cambiar de robot se pueden apreciar en la figura A.4.

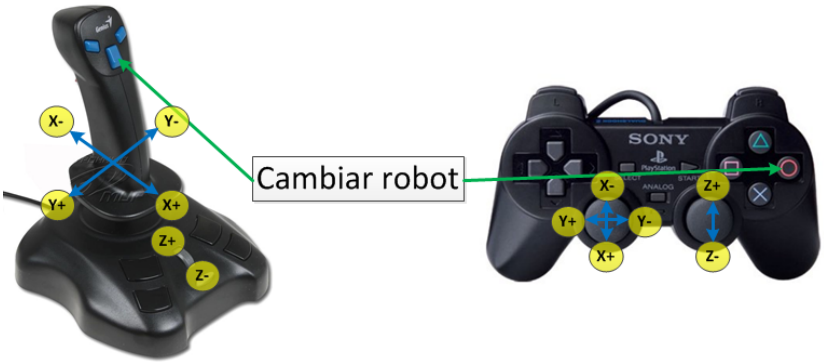

Figura A.4: Funciones y movimiento con el joystick

A.3.4. Realización de una consigna

Con la comunicación serial establecida correctamente, la tarjeta electrónica funcionando y realizando movimientos en los robots, se pueden realizar las consignas geométricas que el software contiene de forma predeterminada.

Para realizar una de las consignas se selecciona en la barra de menú *Consigna* donde se puede ejecutar cualquiera de las consignas preestablecidas: Circular, Lineal, Rombo, Elipse y 3D (ver figura A.5).

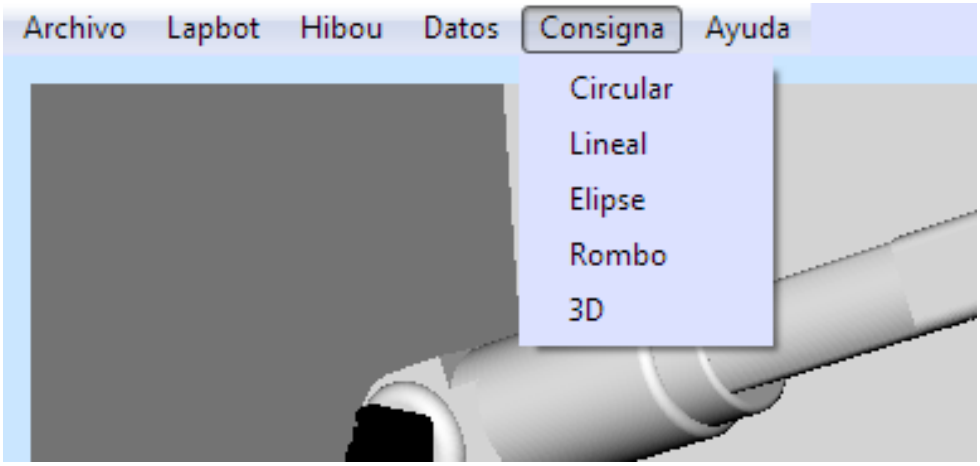

Figura A.5: Consignas disponibles en el menú de RELAS

Al ejecutar cualquier consigna, se muestra un mensaje indicando que el joystick ha sido deshabilitado. (ver figura A.6).

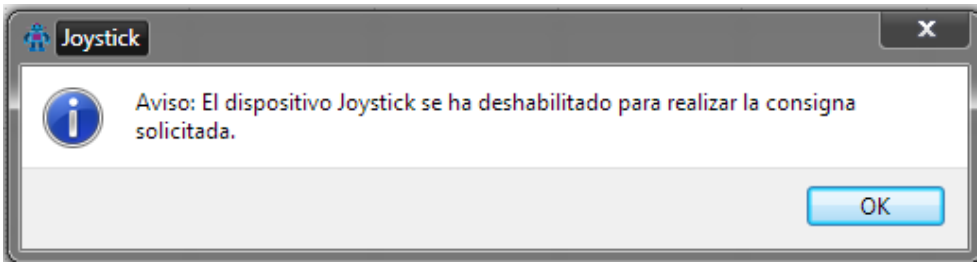

Figura A.6: Realización consigna circular

Después de seleccionar una de las consignas, se mostrará una ventana con la consigna seleccionada. La consigna deseada se graficará en azul, mientras la real en color rojo se empezará a visualizar a medida que esta se vaya realizando (ver figura A.7).

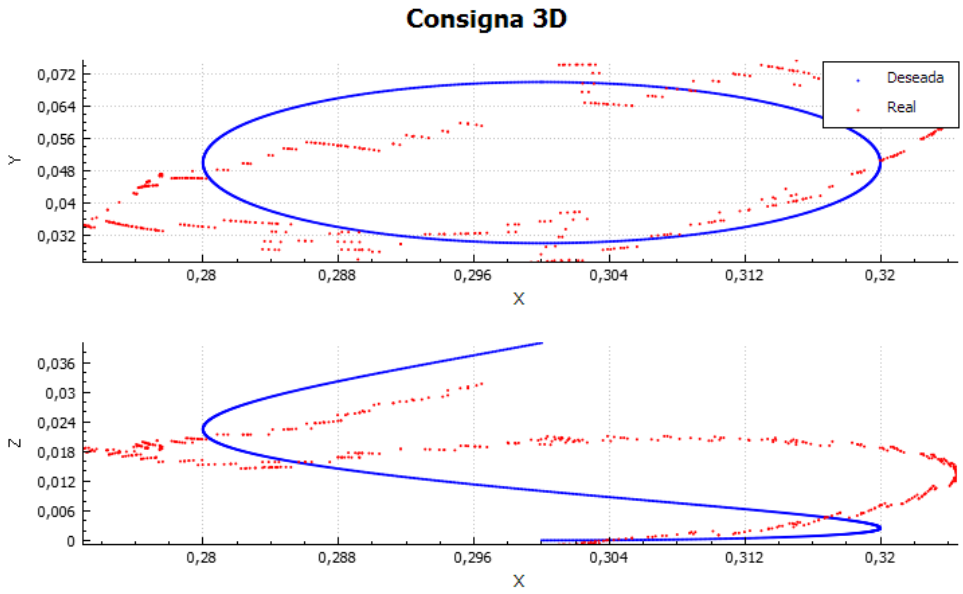

Figura A.7: Consigna 3D en el software

Una vez que la consigna haya finalizado, se debe hacer click en una de las de las consignas geométricas para activar de nuevo el joystick y dar por concluida la consigna. Este método impide que se activen dos consignas al tiempo.

A.3.5. Gráficas de error cartesiano y articular

Los menús desplegables en la barra de menús con las etiquetas Hibou y Lapbot ofrecen la opción de mostrar las gráficas de error cartesiano y articular para cada uno de ellos (Ver figura A.8).

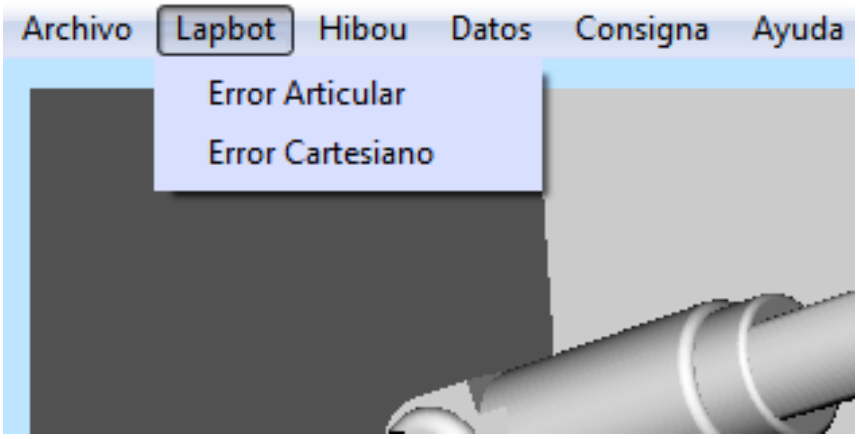

Figura A.8: Menú desplegable para graficar los errores articulares o cartesianos

Al hacer click de acuerdo al robot, una barra se desplegará con las opciones de gráficas. Al seleccionar una de las gráficas, en estas se podrán ver los datos correspondientes graficados, la leyenda si se requiere y sus valores en el eje Y con su respectiva unidad de ingeniería y el tiempo transcurrido en el eje X. El rango de tiempo puede ser cambiado utilizando el control deslizante ubicado en la parte inferior de la ventana, el máximo tiempo es 30 segundos y el mínimo 6 segundos (ver figura A.9).

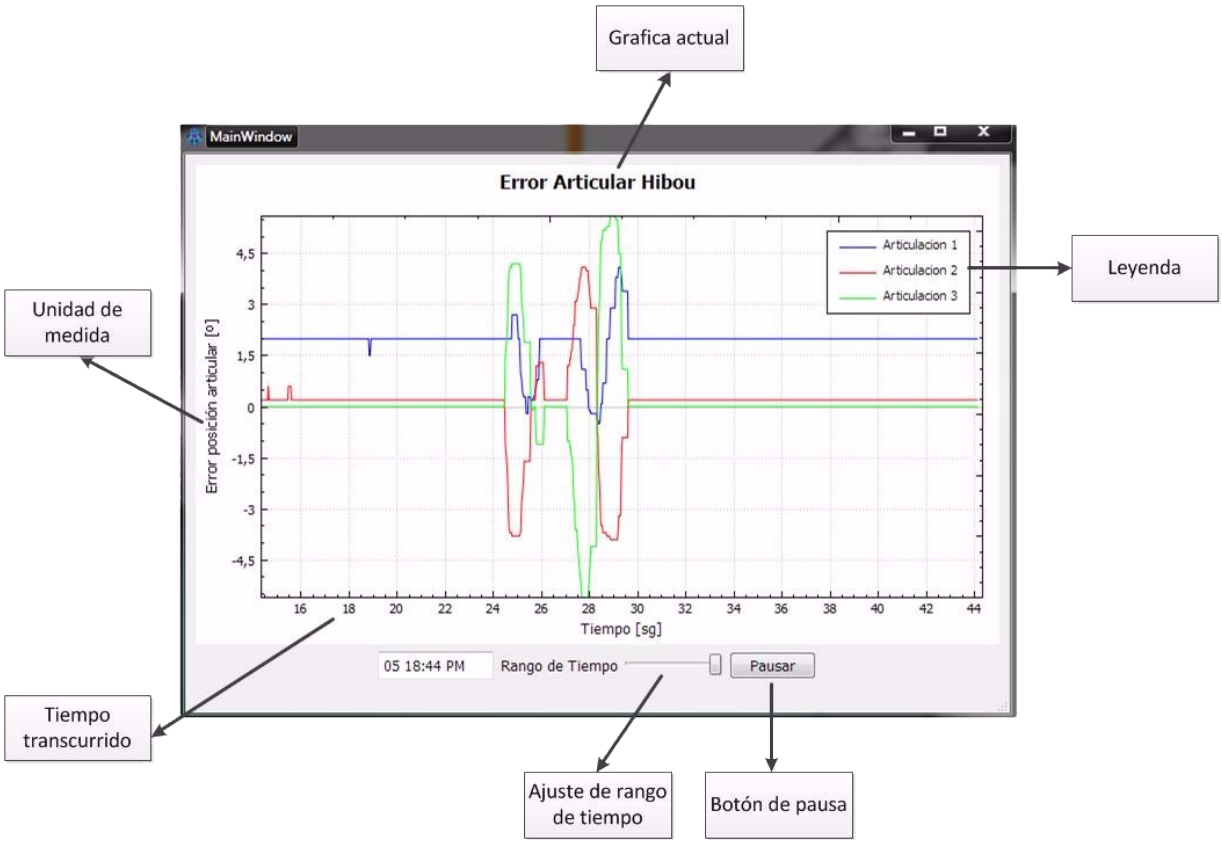

Figura A.9: Ventana de error articular - Hibou

El botón *Pausar* sirve para congelar la figura actual y generar una imagen que será guardada en la carpeta "Capturas de pantalla" en el sitio de instalación del software.

A.3.6. Ventanas de datos numéricos

El Menú Datos ofrece información numérica y precisa del estado de los robots, como es posición real y deseada en grados, posición deseada del joystick en el plano tridimensional y la cadena de datos actualmente recibida en el puerto serial (Ver figura A.10).

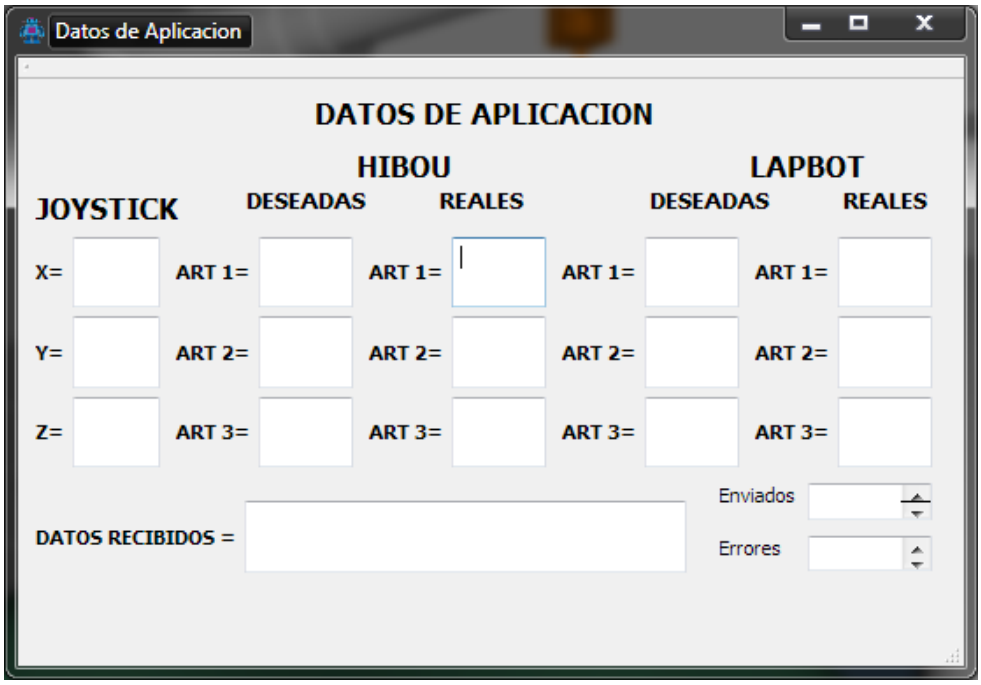

Figura A.10: Datos reales de las posiciones de los robots

A.4. Solución de problemas

La Aplicación no puede ejecutarse

Es posible que su equipo no cuente con todos requisitos de sistema necesarios para el funcionamiento del software RELAS.

Solución 1. Dentro la carpeta "RELAS" sitio de instalación del software, se encuentra una carpeta llamada "Soporte" en ella se encuentra un ejecutable llamado "AIO.exe", Ejecútelo y espere a su finalización, reinicie el equipo y vuelva a abrir el software.

Si luego de realizar el procedimiento descrito en "Solución 1", el problema persiste, por favor reinstale la aplicación.

Aplicación Lenta

Revise que los requisitos *hardware* de su computador cumplan con lo recomendado.

Puerto COM no inicia

- Revise que el dispositivo no se encuentre ocupado, por ejemplo otra aplicación este usando el puerto actual, por ejemplo el *Monitor serial* del *software* Arduino.

- Intente detener el puerto y volverlo a abrir.
- Desconecte y conecte el Arduino de nuevo.
- Cambie el número del puerto COM de Arduino usando las opciones avanzadas en el administrador de dispositivos.

Existe un problema de verificación de firmas de controladores que impide el uso de la tarjeta Arduino, si al verificar el numero COM de su tarjeta Arduino, esta aparece como en la figura A.11.

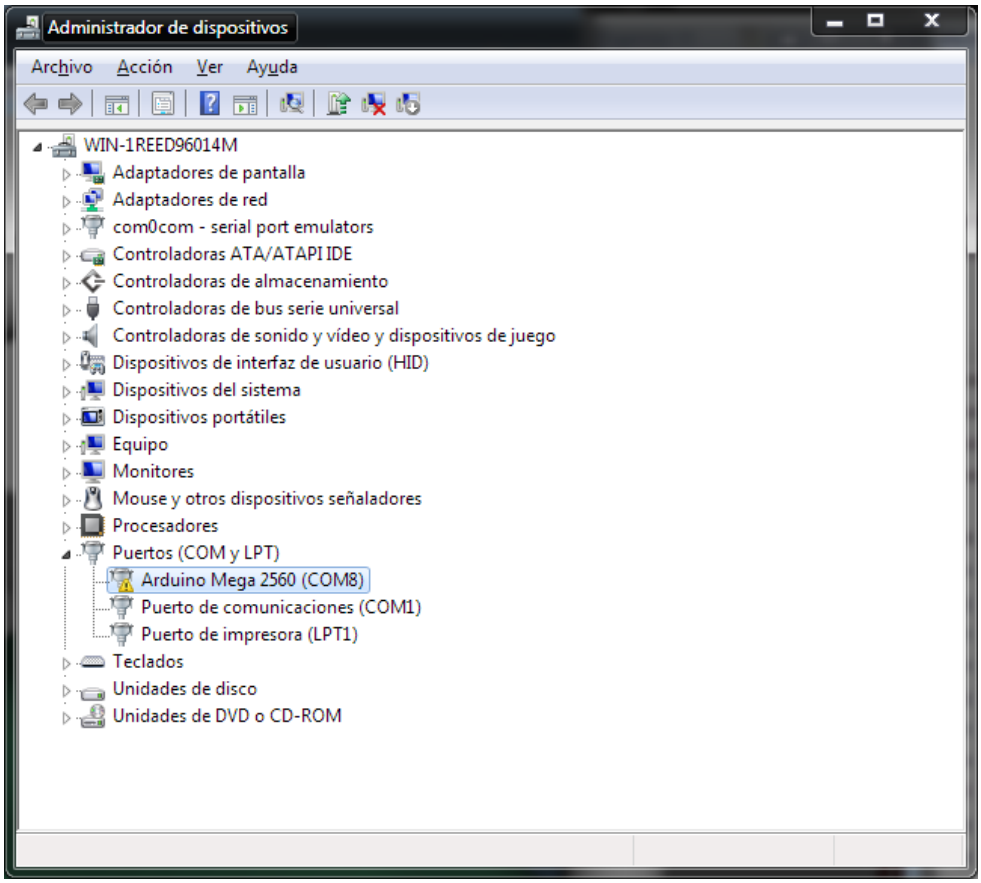

Figura A.11: Administrador de dispositivos

Y al dar "clic derecho, propiedades" en el puerto COM se visualiza un mensaje como el de la figura A.12, entonces debemos reiniciar el computador e iniciarlo en el modo "Opciones de inicio avanzadas".

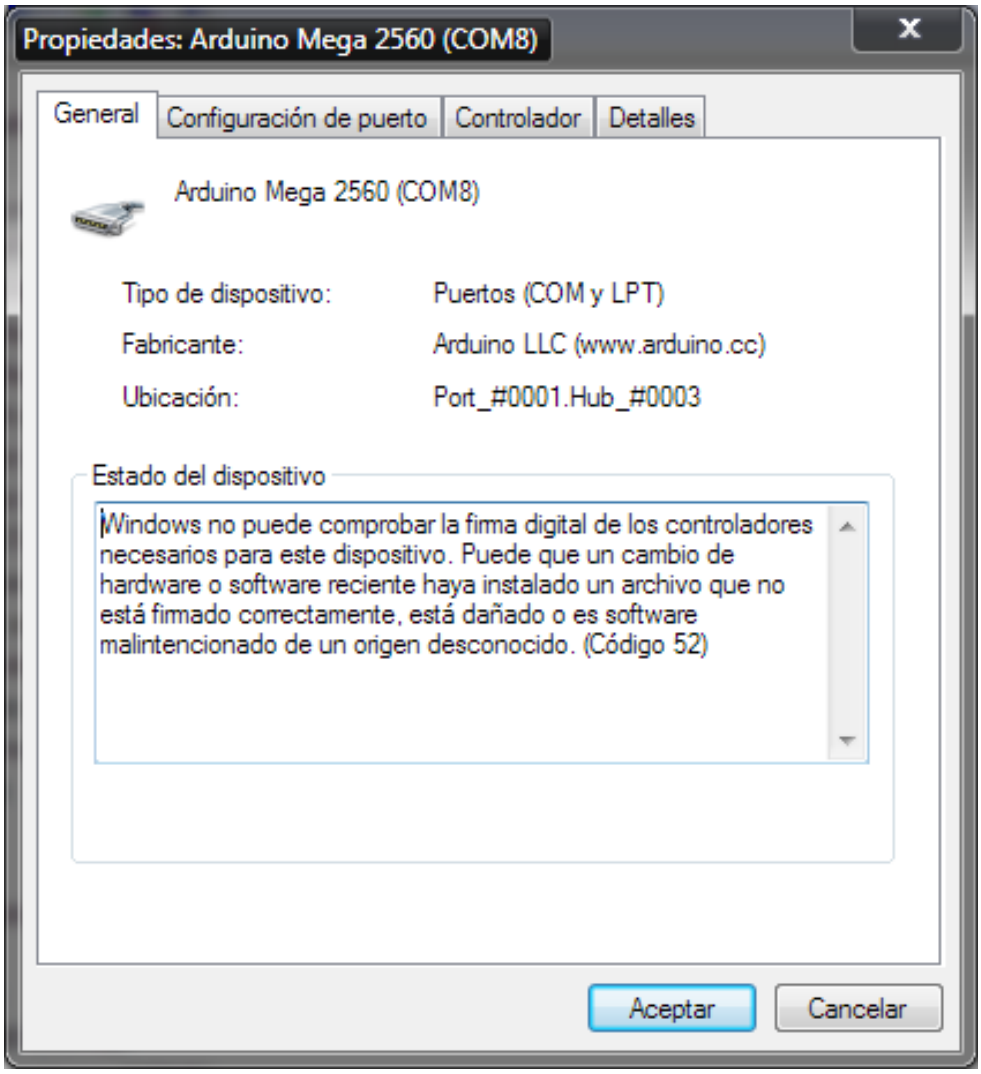

Figura A.12: Propiedades del puerto COM Arduino

Para entrar al modo "Opciones de inicio avanzadas", reinicie el equipo y cuando aparezca el mensaje de la BIOS, presione la tecla F8 hasta que aparezca una ventana como esta A.13.

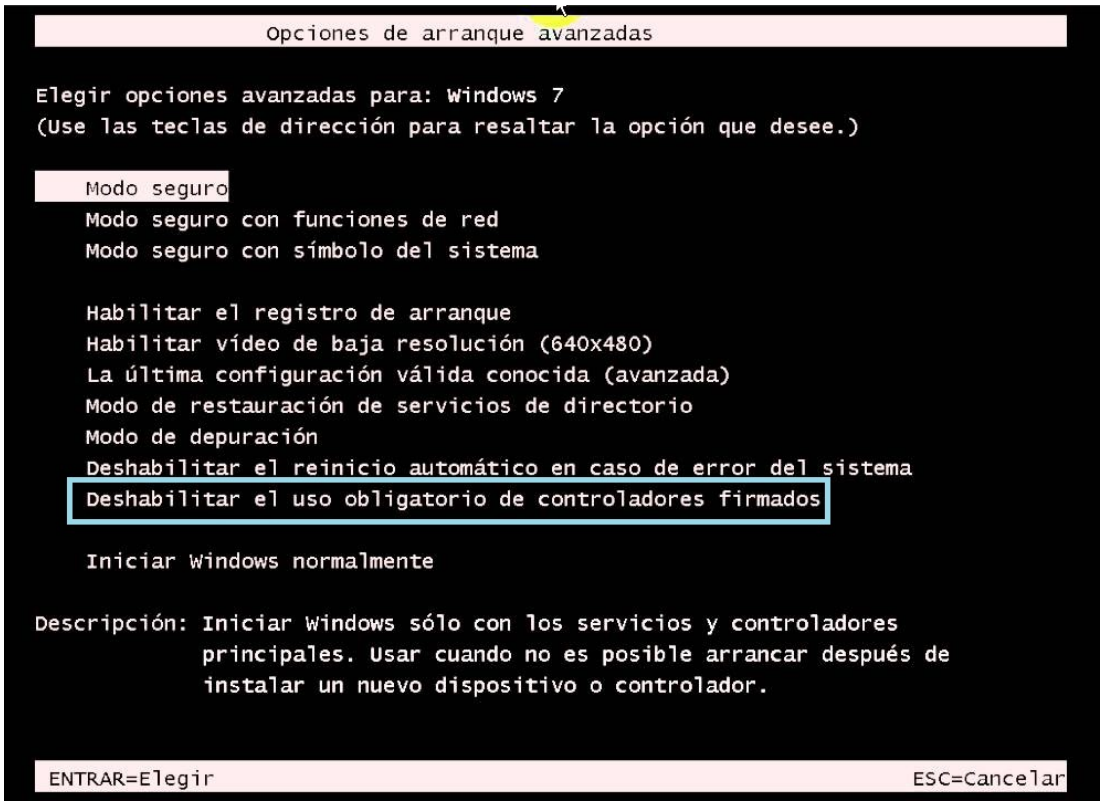

Figura A.13: Opciones de inicio avanzadas

Con las teclas, diríjase a la opción "Deshabilitar el uso obligatorio de controladores firmados" y presione "Enter". Con esto se soluciona el problema y podrá usar su tarjeta Arduino normalmente.

NOTA: Este procedimiento debe realizarse cada vez que se enciende del equipo.