Universidad Nacional de San Luis
Facultad de Ingeniería y Ciencias Agropecuarias

EL TÍTULO SE ESCRIBE EN EL CENTRO SUPERIOR DE LA HOJA Y PODRÁ OCUPAR MÁS DE UN RENGLÓN, EN CUYO CASO TENDRÁ INTERLINEADO DE 1,5 pto.

Autor o autores

Trabajo final de Ingeniería (agregar carrera que corresponda)

Director
Codirector / Codirector Técnico
Asesor/s

Villa Mercedes, San Luis
año

DERECHO DE AUTOR

DEDICATORIA

Texto de dedicatoria justificado.

AGRADECIMIENTOS

Texto de agradecimientos justificado.

RESUMEN

El presente trabajo final de grado tiene como objetivo el diseño y fabricación de un controlador e interfaz gráfica modular para robots antropomorfos colaborativos (COBOTS) de 6 grados de libertad, brazos robóticos de 6 ejes los cuales se pueden programar moviéndolo con la mano indicándole a donde y/o como uno quiere que se mueva, aparte de poder programarlo de la forma habitual. Para poder probar su funcionamiento se armó un robot cuyo diseño de la estructura fue brindado por “ labme “ . Todo el proyecto, controlador mas robot, se pensó para funcione como herramienta didáctica en el Laboratorio de Mecatronica de la Facultad de Ingenieira y Ciencias agropecuarias (LABME). Para la parte principal del controlador se utilizó un sistema operativo para robots utilizado en la industria e investigación llamado ROS, el cerebro del robot, a esto se le añadió una base de datos para guardar bien los puntos y rutinas y se comunicó con un arduino que se encarga de controlar los motores del robot en base a lo que ROS le indique. En cuanto a la interfaz, se realizó una página web donde el usuario puede programar y controlar el robot.Se implementaron 2 modos de uso, uno donde el usuario puede indicarle mediante la interfaz a donde quiere que se mueva, ya sea como punto o armando una rutina de varios puntos ingresados o guardados previamente, y el otro modo donde el usuario puede manipular el robot manualmente y guardar posiciones o toda una trayectoria completa para luego correrla o modificarla con el otro modo. Se realizaron pruebas para validar la operatividad y cumplimiento de los objetivos. Los resultados obtenidos demostraron el correcto funcionamiento del controlador e interfaz. Estas experiencias validaron su funcionalidad como herramienta didáctica, proporcionando a los estudiantes experiencia práctica en robótica y control, abriendo la posibilidad de futuras ampliaciones y mejoras del sistema.

El resumen debe ser de un solo párrafo, escrito en lenguaje apto para todo público sin utilizar terminología técnica específica. Debe ser un reflejo preciso del contenido del documento. No contendrá ecuaciones matemáticas, citas de referencia ni notas al pie. Su extensión debe ser de al menos 500 palabras y no más de una carilla. Fuente Arial tamaño 11, justificado, interlineado 1.5.

El presente trabajo final de grado tiene como propósito desarrollar un sistema completo que permita manejar y programar un brazo robótico de manera sencilla, pensado uso didáctico. Para ello, se diseñó y construyó un controlador y una interfaz capaces de operar un robot colaborativo, es decir, un tipo de robot que puede ser guiado directamente con la mano por el

usuario para indicarle los movimientos que se desean realizar. Con el fin de evaluar el funcionamiento del sistema desarrollado, se ensambló un robot utilizando un diseño estructural proporcionado por el laboratorio universitario donde se llevó a cabo este proyecto. Tanto el robot como el controlador fueron concebidos como herramientas educativas para el Laboratorio de Mecatrónica de la Facultad de Ingeniería y Ciencias Agropecuarias, con el objetivo de ofrecer a los estudiantes un equipo accesible que facilite la comprensión práctica de los conceptos básicos de control y movimiento de robots. El sistema creado permite trabajar de dos maneras principales. Por un lado, el usuario puede indicar mediante la interfaz a qué posición desea que el robot se desplace, ya sea cargando posiciones previamente guardadas o ingresando nuevas, lo que posibilita planificar tareas paso a paso. Por otro lado, el robot también puede ser movido directamente con la mano, permitiendo que el propio usuario le muestre físicamente el recorrido o las posiciones que desea repetir. El sistema registra estos movimientos y puede reproducirlos posteriormente, lo que lo convierte en una herramienta muy útil para la enseñanza, ya que permite experimentar de forma directa con diferentes formas de programar un robot. La interfaz desarrollada, basada en una página web accesible desde una computadora, hace posible que cualquier usuario pueda manejar el robot sin necesidad de conocimientos previos, ofreciendo una experiencia clara y ordenada. Finalmente, se realizaron diversas pruebas para comprobar el correcto funcionamiento tanto del controlador como del robot y la interfaz. Los resultados obtenidos mostraron que el sistema cumple con los objetivos planteados, permitiendo controlar el robot de manera fluida y registrar distintos tipos de movimientos para su posterior uso. Además, el proyecto demostró su utilidad como herramienta didáctica, ya que facilita que los estudiantes puedan interactuar con el robot, comprender su comportamiento y experimentar con diferentes formas de indicarle movimientos. El sistema desarrollado sienta las bases para futuras mejoras y ampliaciones, permitiendo que nuevas generaciones de alumnos continúen explorando y aprendiendo a partir de un recurso práctico y adaptable a distintos niveles de formación

Palabras claves — Cobot, Robot antropomórfico, ROS.

.

ÍNDICE DE CONTENIDO

CAPITULO 1: Propuesta 8

Introducción 8

Objetivos 9

Objetivo general 9

Objetivos específicos 9

Alcances y limitaciones 10

Marco teórico y/o justificación y/o estado del arte 11

Marco teórico 11

Justificación 11

Estado del arte 11

CAPITULO X: Análisis y Desarrollo 12

CAPITULO X: Análisis de Costos 13

CAPITULO X: Estudio de Impacto Ambiental 14

CAPITULO X: Conclusiones 15

Glosario 16

Referencias Bibliográficas 17

Anexo/s 18

Apéndice/s 19

ÍNDICE DE FIGURAS

Figura Nº 1. Esquema de experiencia sobre polarización. 12

ÍNDICE DE TABLAS

Tabla Nº1. Abreviaturas 12

CAPITULO 1: Propuesta

Introducción

Texto de la introducción. Texto de la introducción.

Objetivos

Objetivo general

- Diseñar y desarrollar un sistema de control y una interfaz para un robot colaborativo de tipo antropomórfico, capaces de operar en dos modos principales —“movimiento” y “lectura”—, integrando el hardware necesario en un robot construido para la validación práctica del sistema y evaluando su potencial como herramienta didáctica en el ámbito universitario.

Objetivos específicos

- Definir y seleccionar los componentes electrónicos y de control necesarios para el funcionamiento del sistema, incluyendo actuadores, sensores, microprocesadores y elementos de comunicación.
- Integrar el sistema de control con la estructura del robot disponible en el LABME, realizando el montaje del hardware indispensable para su operación y pruebas
- Desarrollar la arquitectura de software del controlador, asegurando la comunicación entre los distintos dispositivos, el procesamiento de datos y la respuesta adecuada del sistema
- Implementar los modos de operación “movimiento” y “lectura”, creando los procedimientos necesarios para registrar posiciones, reproducir trayectorias y ejecutar movimientos indicados por el usuario.
- Seleccionar e implementar la base de datos adecuada para el proyecto, integrándola al sistema y desarrollando el módulo encargado de gestionar las funciones de consulta, registro y modificación de la información.
- Diseñar y programar una interfaz web que permita interactuar de forma sencilla con el robot, contemplando funciones de programación, control y visualización.
- Ensamblar y ajustar el robot utilizado como plataforma de pruebas, garantizando su correcto funcionamiento mecánico y electrónico.

- Realizar pruebas individuales de cada módulo y pruebas de integración del sistema completo, identificando mejoras necesarias y optimizando el rendimiento general.
- Evaluar el desempeño del sistema desarrollado en escenarios representativos del uso didáctico, analizando su facilidad de uso, exactitud, seguridad y capacidad para apoyar actividades de enseñanza.

Alcances y limitaciones

El proyecto abarca el diseño y desarrollo del controlador de un robot colaborativo de seis grados de libertad, así como el montaje y puesta en funcionamiento del robot utilizado como plataforma de validación. Incluye además la implementación de una interfaz de usuario basada en una página web que permite programar, registrar y reproducir movimientos y trayectorias. El alcance cubre la integración del hardware necesario, el desarrollo del software de control y la realización de pruebas funcionales que aseguren el correcto desempeño del sistema.

La aplicación del sistema se encuentra limitada exclusivamente al ámbito didáctico, académico y de experimentación dentro del laboratorio universitario. Debido a los componentes empleados y a los objetivos del proyecto, el robot y su controlador no están destinados a cumplir con los requisitos de precisión, robustez, seguridad o capacidad de carga propios de aplicaciones industriales.

Marco teórico y/o justificación y/o estado del arte

Marco teórico

En esta sección se consignará el contenido del marco teórico.

ROS

Archivo .xacro?

Moveit?

Pilaz?

React

Encoders

Dynamixels

Cobots

Base de datos

Arduinp+Librerias?

Justificación

En esta sección se consignará el contenido de la justificación.

Este proyecto forma parte de una iniciativa mayor del Laboratorio de Mecatrónica orientada a incorporar robots didácticos que representen modelos utilizados en la industria. El desarrollo del software de control para un cobot de 6 grados de libertad contribuye a ampliar las herramientas disponibles para la formación práctica de los estudiantes. Si bien el hardware acompaña el proyecto con fines demostrativos, el eje principal de este trabajo final es el diseño

e implementación del sistema de control, fundamental para brindar a los alumnos una experiencia más cercana a la operación y programación de robots industriales.

El presente proyecto se enmarca dentro de una iniciativa mayor del Laboratorio de Mecatrónica destinada a incorporar plataformas robóticas didácticas que representen, a escala, los sistemas utilizados en la industria. En este contexto, el desarrollo de un controlador para un robot colaborativo de seis grados de libertad contribuye a ampliar las herramientas disponibles para la enseñanza y la experimentación.

Si bien el eje principal del trabajo es la implementación del software de control, se integra también un prototipo físico del robot a fin de disponer de una plataforma funcional que permita validar el sistema y que, a futuro, pueda ser mejorada o utilizada en otras líneas de trabajo. De esta manera, el proyecto aporta tanto a la formación académica como al fortalecimiento tecnológico del laboratorio

Estado del arte

En esta sección se consignará el contenido del estado del arte.

CAPITULO X: Análisis y Desarrollo

2. Análisis y Desarrollo
    2.1 Presentacion de las partes del sistema
        2.1.1 Robot
            2.1.1.1 Estructura
            2.1.1.2 Actuadores
            2.1.1.3 Placas Electronica
            2.1.1.4 Arduino
        2.1.2 Base de Datos
            2.1.2.1 Tinydb
            2.1.2.2 Modulo db_puntos.py
        2.1.3 Arquitectura ROS
            2.1.3.1 modo_lectura
            2.1.3.2 modo_control
            2.1.3.3 controlador_cobot
            2.1.3.4 publicador_posicion_real
            2.1.3.5 Archivo inicializador launch
        2.1.4 Interfaz Grafica
    2.2 Comunicacion o intercomunicación
    2.3 Manual de uso
        2.3.1 Modo control
        2.3.2 Modo lectura
            2.3.2.1 modo punto
            2.3.2.2 modo trayectoria
    2.4 Manual de instalacion

2.1 Presentación general del sistema

- Excelente idea.
- Aquí explicás brevemente qué partes tiene el sistema (robot, controlador ROS, Arduino, base de datos, interfaz), pero SIN meterte en detalles todavía.
- Es como un mapa general.

2.1.1 Robot

Subdividido en:

- Estructura
- Actuadores
- Electrónica
- Arduino

Perfecto. No es redundante si en 2.1 hacés solo un “pantallazo general”.

2.1.2 Base de datos (TinyDB)

Correcto.
Deberías incluir:

- Por qué TinyDB y no SQLite, PostgreSQL, etc.
- Cómo se organiza la información (rutas, puntos, metadatos).

2.1.3 Arquitectura ROS

Muy importante en tu tesis.
Aquí deberían ir:

- Explicación de la arquitectura ROS que diseñaste
- Función de cada nodo
- Qué hace cada servicio / tópico
- Cómo se comunican
- Por qué elegiste esa estructura

Los subapartados que pusiste están buenísimos.

2.1.4 Interfaz gráfica

Perfecto que expliques la evolución y diseño final.
También podés incluir:

- Interacción con ROS
- Tecnologías usadas (sin profundizar demasiado)

2.2 Comunicación o Intercomunicación

Esta sección es **clave**.
Aquí explicás cómo encaja TODO:

- Arduino ↔ ROS
- ROS ↔ Web
- ROS ↔ Base de datos
- Usuario ↔ Interfaz

También describís:

- Cada tópico
- Qué publica, qué recibe
- Frecuencias
- Intercambio de mensajes
- Arquitectura general del sistema

Esta sección no debe eliminarse.
Es lo que muestra que el sistema es coherente.

2.3 Manual de uso

Totalmente válido.
Muchas tesis de ingeniería lo incluyen al final del capítulo técnico.

2.4 Manual de instalación

También es completamente válido.
De hecho, **es un aporte enorme** para que otros estudiantes puedan replicarlo.
Incluye:

- Instalación de ROS
- Dependencias
- Estructura de carpetas
- Permisos y configuraciones
- Iniciar los nodos
- Iniciar la interfaz web
- Conectar Arduino

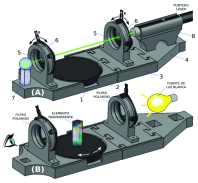
. En este capítulo se presenta el análisis y el diseño de los componentes que conforman el sistema —el robot, el controlador y la interfaz— así como la forma en que se organizan e interactúan entre sí. Cada uno de estos elementos cumple una función específica dentro del conjunto. A continuación, se describen en detalle sus características principales, su estructura y su comportamiento dentro del sistema.

Antes de avanzar con el análisis particular de cada parte, es necesario destacar algunos criterios considerados durante el proceso de diseño. Desde el inicio, el proyecto se planteó con un enfoque modular y escalable, de manera que cada componente pudiera modificarse, reemplazarse o analizarse de forma independiente sin afectar el resto del sistema. Asimismo, se priorizó que la solución desarrollada pudiera migrarse fácilmente a una Raspberry Pi, permitiendo prescindir de una computadora externa y facilitando su uso en entornos educativos o demostrativos..

2.1 Robot

El robot está compuesto por una estructura impresa en 3D, cuatro actuadores Dynamixel modelo XL-430, tres actuadores Dynamixel modelo XL-320, una placa controladora Arduino ATmega2560, dos módulos de “BOTONERA”, una placa controladora dedicada a los motores, y una fuente de alimentación principal de 11.1 V. En la Figura: se presenta una vista general del robot con la identificación de cada uno de sus componentes

[FIGURA X]

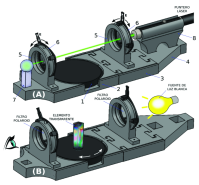


Estructura

El diseño estructural del robot fue desarrollado por **Gabriel Iglesias**, y se basa en una configuración típica de robots colaborativos, similar a la empleada por fabricantes industriales, por su estabilidad mecánica y facilidad para resolver la cinemática Figura:, (REFERENCIA). Inicialmente, los eslabones correspondientes al brazo desde el hombro hasta la muñeca tenían

una longitud mayor, pero se optó por reducirla debido a que, durante movimientos amplios, se generaban cargas elevadas sobre los actuadores del hombro. Esto producía incrementos de corriente que activaban los mecanismos de protección interna de los motores, ocasionando detenciones o desconexiones durante el funcionamiento.

[FIGURA X]



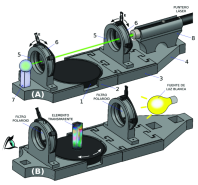
Actuadores

En cuanto a los actuadores, se emplearon motores Dynamixel XL-430 (Figura:) y XL-320 (Figura:). Estos componentes habían sido adquiridos por el LabMe con fines de investigación, por lo que resultaron adecuados para el presente proyecto. En un inicio, el sistema utilizaría únicamente actuadores XL-430 en todas las articulaciones; sin embargo, se decidió reemplazar los tres actuadores distales por XL-320 debido a que estas articulaciones requieren menor torque. Este cambio permitió reducir la carga total soportada por los actuadores proximales, posibilitando un brazo de mayor alcance sin comprometer la seguridad ni la precisión.

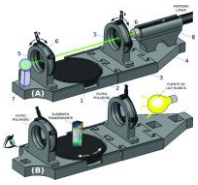
Este reemplazo requirió modificaciones adicionales: los actuadores XL-430 operan óptimamente a 11.1 V, mientras que los XL-320 requieren 7.4 V y utilizan conectores diferentes. Para resolver esta incompatibilidad se incorporó un integrado “NOMBRE DE INTEGRADO”, encargado de convertir la tensión de alimentación de 11.1 V a 7.4 V y permitir la conexión eléctrica entre ambos tipos de actuadores.

Aun así, el actuador del hombro continuaba presentando limitaciones en movimientos de gran amplitud debido al peso acumulado de los motores distales. Para mitigar este problema, se incorporó un segundo motor sincronizado en dicha articulación, aumentando la capacidad de torque útil y reduciendo los errores durante el movimiento. Posteriormente, esto se complementó con el acortamiento de los eslabones previamente mencionado.

[FIGURA X]



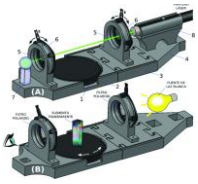
[FIGURA X]



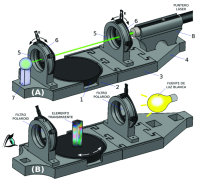
Placas Electronica

Considerando la futura migración del sistema ROS hacia una Raspberry Pi, se incorporó una “BOTONERA” destinada a permitir el control básico del robot sin depender de la interfaz gráfica. Esto es especialmente útil para demostraciones rápidas, pruebas de laboratorio o exposiciones. La BOTONERA se compone de dos módulos: uno ubicado en la base del robot, que permite seleccionar modos de operación y ejecutar la última rutina almacenada (Figura:), y otro ubicado en el eslabón entre el codo y la muñeca, utilizado para desenclavar los motores y registrar puntos (Figura:). Su funcionamiento detallado se describe más adelante.

[FIGURA X]

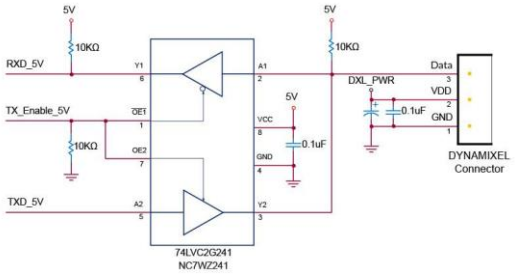


[FIGURA X]

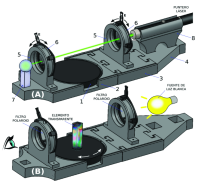


Además, se diseñó una tercera placa destinada al control de los actuadores Dynamixel (Figura:), la cual permite reutilizar el sistema de control en proyectos futuros. (REFERENCIA?)

[FIGURA X]



[FIGURA X]

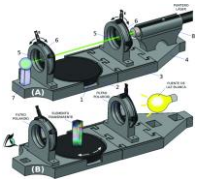


Arduino

Por último, en la base del robot se integraron la placa controladora de Dynamixel, el módulo principal de la BOTONERA y el controlador Arduino ATmega2560, todos conectados a la fuente de 11.1 V y con punto de neutro común (Figura:). El Arduino cumple las siguientes funciones:

- **Comunicación con los actuadores Dynamixel:** mediante la "LIBRERÍA DYNAMIXELS", el Arduino envía las posiciones objetivo a los motores y obtiene la posición real a una frecuencia de 0.5 s.
- **Comunicación con ROS e interfaz web:** utilizando la librería "ROSERIAL", el Arduino actúa como un nodo de ROS, recibiendo los puntos generados desde el sistema y enviando la telemetría (posiciones reales y estados de la BOTONERA) tanto a ROS como a la página web,
- **Gestión de la BOTONERA:** procesa los eventos y comandos provenientes de los módulos físicos de control manual.".

[FIGURA X]



2.2 **Base de Datos**

Para este proyecto se consideró desde el inicio la posibilidad de migrar el controlador a una Raspberry Pi. Por este motivo, se buscó una base de datos ligera, simple y con buen rendimiento, capaz de funcionar correctamente en entornos de recursos limitados y compatible con el sistema operativo utilizado. Además, se evaluaron aspectos como la facilidad de integración con Python y la forma en que el sistema produce y consume los datos.

Para la elección final se consultó a un colega con amplia experiencia profesional en bases de datos y en sistemas Linux. Luego de presentarle los criterios de diseño y el uso previsto dentro del controlador, recomendó emplear TinyDB [REFERENCIA], una base de datos NoSQL basada en archivos JSON, de bajo consumo de recursos y muy adecuada para aplicaciones de tamaño medio que requieren simplicidad y rapidez de implementación.

db_puntos6.py

Con el fin de facilitar el acceso a la base de datos y mantener el software organizado, se desarrolló un archivo independiente que funciona como intermediario entre el sistema y TinyDB. Este módulo centraliza todas las funciones necesarias para leer, almacenar, modificar y eliminar información, proporcionando una interfaz clara y uniforme para el resto del proyecto. Gracias a esta separación, los demás componentes no necesitan interactuar directamente con TinyDB, lo que simplifica el desarrollo, evita duplicación de código y mejora la mantenibilidad y escalabilidad del sistema.

2.3 Arquitectura ROS:

En este proyecto, ROS se utiliza como la plataforma principal para organizar, coordinar y comunicar los distintos componentes del sistema. Los nodos desarrollados en Python interactúan entre sí mediante tópicos, servicios y acciones, lo que permite estructurar el controlador del robot de manera modular y distribuida.

Si bien algunos nodos realizan funciones directamente relacionadas con ROS —como el cálculo de trayectorias, la ejecución de movimientos a través de MoveIt/Pilz o la obtención de la cinemática— otros cumplen tareas adicionales, como gestionar la comunicación con Arduino, interactuar con el módulo de la base de datos o procesar la información proveniente de la página web. Estos procesos no forman parte del núcleo de ROS, pero se integran dentro de nodos para mantener una arquitectura unificada y facilitar la comunicación entre todos los módulos del sistema.

A continuación, se describen los nodos que conforman el controlador y las funciones principales que realiza cada uno.

modo_lectura (modo_lectura18.py)

El nodo modo_lectura actúa como el administrador central de toda la información del sistema. Se encarga de recibir los datos reales enviados por Arduino —que corresponden a las posiciones del robot y ordenes de la "BOTONERA"—, procesarlos, convertirlos a formatos utilizables y enviarlos a la interfaz web para su visualización. Al mismo tiempo, cuando la interfaz web realiza alguna acción (por ejemplo agregar, modificar o eliminar puntos o rutinas), este nodo interpreta esos comandos y se comunica con el módulo encargado de la base de datos para guardar, actualizar o recuperar la información solicitada y posteriormente darle una devolución a la interfaz para corroborar los cambios o informar fallos.

modo_control (trayectoria37.py)

Este nodo es el encargado de coordinar el movimiento del robot. Recibe las órdenes para ejecutar una rutina completa o una trayectoria simple, ya sea desde la página web o desde la "BOTONERA". A partir de estas órdenes, consulta los puntos almacenados en la base de datos o utiliza la información enviada por la página web para trayectorias simples. Con estos datos arma la secuencia correspondiente y se la envía al planificador de MoveIt (Pilz), que genera los planes de movimiento del brazo robótico de acuerdo con las indicaciones establecidas, ya sea en términos de velocidad, precisión o tipo de trayectoria. Además, administra las pausas programadas. De esta manera, el nodo funciona como el "centro de control" que organiza, gestiona y ejecuta todos los movimientos del robot

controlador_cobot (controlador_rutina_prueba4.py)

El nodo controlador_cobot cumple el rol de coordinador de la ejecución de trayectorias generadas por MoveIt. Su función principal es interceptar los planes de movimiento enviados por el nodo de planificación, validar su estado mediante el feedback de MoveIt/Pilz y, una vez verificados, transformar y enviar los puntos articulares al Arduino para que el robot los ejecute físicamente. Además, interpreta si debe ejecutarse una trayectoria calculada por MoveIt o una trayectoria bruta guardada por el usuario (sub-modo trayectoria), y administra los tiempos de espera definidos en la rutina.

Este nodo trabaja en conjunto directo con el nodo modo_trayectoria, que es el encargado de solicitar a MoveIt la generación de los planes. Una vez que modo_trayectoria inicia una planificación, controlador_cobot recibe la señal de control correspondiente, espera la llegada de los fragmentos del plan junto con el feedback de estado (“PLANNING”, “MONITOR”, “IDLE”), y determina si la planificación fue válida. En caso afirmativo, ejecuta el plan completo enviando los puntos articulados ya convertidos al formato requerido por los servomotores Dynamixel. Durante la ejecución informa a modo_trayectoria el avance mediante mensajes específicos, indicando si debe enviar el siguiente movimiento, si la ejecución fue correcta o si ocurrió un error.

De esta manera, controlador_cobot actúa como el módulo responsable de supervisar, validar y ejecutar los movimientos del robot físico, garantizando que solo se ejecuten trayectorias correctas y administrando eventos adicionales como pausas programadas y trayectorias capturadas manualmente. Es el nexo entre el planificador de alto nivel (MoveIt) y el actuador final (Arduino), asegurando que la ejecución siga el orden y la estructura definidos en las rutinas programadas.

publicador_posicion_real (publicacion_posicion_real3.py)

Este nodo recibe las posiciones de los motores enviadas por el Arduino, organiza esos datos en un mensaje adecuado y los publica para que el sistema de control (MoveIt) conozca en todo momento la posición real del robot. Esto permite que los cálculos de movimiento se realicen siempre en base al estado actual del brazo. Aunque esta tarea podría ejecutarse directamente en el Arduino, se decidió trasladarla a ROS para reducir la carga de procesamiento sobre el microcontrolador y evitar demoras, ya que constituye el principal cuello de botella del sistema

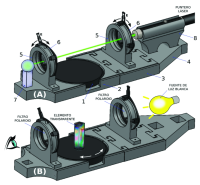
2.4 Interfaz Grafica

Para la interfaz gráfica se decidió desarrollar una página web accesible mediante red LAN, considerando que en el futuro el sistema será migrado a una Raspberry Pi. De esta forma, la interfaz puede visualizarse desde cualquier dispositivo conectado a la misma red, ya sea una PC, notebook, tablet o teléfono.

En una primera etapa, la página fue programada de forma tradicional utilizando HTML, CSS y JavaScript, gestionando sus propios estados internos y almacenando temporalmente la

información del usuario. Por ejemplo, la “Rutina Actual” existía únicamente en la página: si el usuario actualizaba la vista o cerraba la pestaña, todos los datos se perdían, y solo se comunicaba con la Arquitectura ROS al guardar o ejecutar una rutina. Este enfoque dio lugar a la primera versión de la interfaz [FIGURA X].

[FIGURA X]



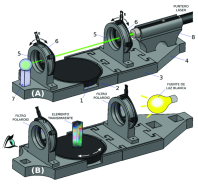
Sin embargo, durante la implementación del control de errores surgieron diversas limitaciones. En primer lugar, se evidenció que la página no podía verificar si una instrucción era válida hasta el momento de ejecutarla, lo que podía generar rutinas con puntos incorrectos o imposibles de realizar. Esto, además, dificultaba que el usuario mantuviera el hilo de lo que estaba programando, especialmente en rutinas extensas, o lo obligaba a ejecutar cada punto individualmente para comprobar su validez. En segundo lugar, se buscó que la “Rutina Actual” fuera más robusta y quedara almacenada en la base de datos, en lugar de depender únicamente del estado temporal de la página, aunque siguiera siendo un dato efímero que se reinicia junto con el controlador. También se identificaron funciones que mejorarían significativamente la comodidad del usuario, como la eliminación de múltiples puntos. Finalmente, se reconoció la necesidad de manejar información altamente dinámica —como la posición en tiempo real del robot—, lo cual requería una plataforma y un enfoque más adecuados.

Para resolver estas limitaciones se replanteó por completo el funcionamiento de la interfaz. La página pasó a actuar como un visualizador del estado real del sistema. Es decir, la página envía una orden, el sistema la procesa y, una vez validada y ejecutada, devuelve a la interfaz los datos que deben mostrarse, aunque coincidan con los que el usuario haya ingresado. De esta manera se garantiza coherencia, seguridad y sincronización entre la interfaz y el controlador.

Como parte de esta reestructuración se decidió reescribir toda la interfaz en React, lo que permitió mejorar la dinámica visual, simplificar la implementación de funciones complejas y

estructurar el código de forma modular, facilitando futuras expansiones del sistema. El resultado de esta mejora se observa en la segunda versión de la interfaz [FIGURA X].

[FIGURA X]

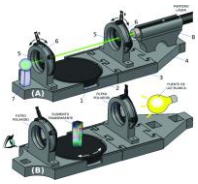


2.2 Conexiones

En este proyecto, el framework **ROS** actúa como la plataforma central para organizar, coordinar y comunicar los distintos componentes del sistema: el robot, los nodos de control, y la interfaz web. Cada nodo cumple una función específica, pero su comportamiento depende de la información que intercambian entre sí mediante **tópicos**, **servicios** y **acciones**. Adicionalmente, la **base de datos** se integra al sistema mediante un módulo que interactúa directamente con los nodos de ROS, permitiendo almacenar y recuperar información relevante para la operación del robot.

Si bien no profundiza en la implementación interna de cada conexión, se presenta una visión general del flujo de comunicación que permite comprender cómo se integra cada parte del sistema. En la [FIGURA] se muestra de manera esquemática la interacción entre los nodos de ROS, el microcontrolador Arduino y la interfaz web, representando el recorrido de la información dentro del sistema.

[FIGURA X]



Con el fin de ofrecer una referencia precisa para futuros desarrolladores y estudiantes —especialmente quienes deseen modificar, extender o reemplazar algún módulo—, en la [TABLA:] se detalla cada tópico utilizado. Para cada uno se especifica el tipo de mensaje, su función dentro del sistema y los nodos que actúan como publicadores o suscriptores. Esta

información permite comprender de forma directa qué influencia tiene cada tópico, dónde se origina cada dato y qué componentes dependen de él

Topico	Tipo de msg	Para que se usa	Publicador	Subscriptor
guardar_trayectoria	Bool	Bandera para guardar una Trayectoria	arduino	modo_lectura
/move_group/display_planned_path	DisplayTrajectory	Plan generado por MoveIt	Moveit	modo_control
				controlador_cobot
/sequence_move_group/feedback	MoveGroupSequenceActionFeedback	feedback de MoveIT indica si se generó bien el plan o hubo error	Moveit	controlador_cobot
cord_dy	Int16MultiArray	Coordenadas a donde se tienen que mover los motores	controlador_cobot	arduino
cord_ros	punto_correr	informacion de punto a ejecutar individualmente	Pagina Web	modo_control
informe_web	msg_web	Mensajes de todos los procesos realizados	Pagina Web	Pagina Web
			modo_lectura	
joint_states	JointState	Coordenadas de la posicion del robot simulado por MoveIT	publicador_posicion_rea	Moveit
lista_puntosdb	nombresPuntos	Lista de Puntos Guardados	modo_lectura	Pagina Web
lista_rutinasdb	nombresPuntos	Lista de Rutinas Guarddas	modo_lectura	Pagina Web
modo_actuar	Bool	Indicador del modo del robot(Control-Lectura)	modo_lectura	modo_lectura
			Pagina Web	Pagina Web
			arduino	arduino
orden_web	punto_web	manda la ordende la pagina web y la informacion necesaria	Pagina Web	modo_lectura
p_dy	Int16MultiArray	coordenadas de puntos guardados	arduino	modo_lectura
planificacion_A_trayectoria	Int16	comando para planificacion de ejecuacion rutinas y puntos entre nodos "controlador_cobot" y "modo_control"	controlador_cobot	modo_control
pos_dy	Int16MultiArray	coordenadas de la posicion real desde el robot	arduino	modo_lectura
				publicador_posicion_rea
pos_real	punto_real	coordendas convertidas para mostrar de la posicion real	modo_lectura	Pagina Web
puntodb	punto_web	informacion de puntos la base de datos para mostrar en la pagina web	modo_lectura	Pagina Web
puntoRutina	punto_web	informacion de rutinas la base de datos para mostrar en la pagina web	modo_lectura	Pagina Web
rutina	Bool	bandera para ordenar la ejecución de la rutina	modo_control	modo_control
			arduino	arduino
			Pagina Web	

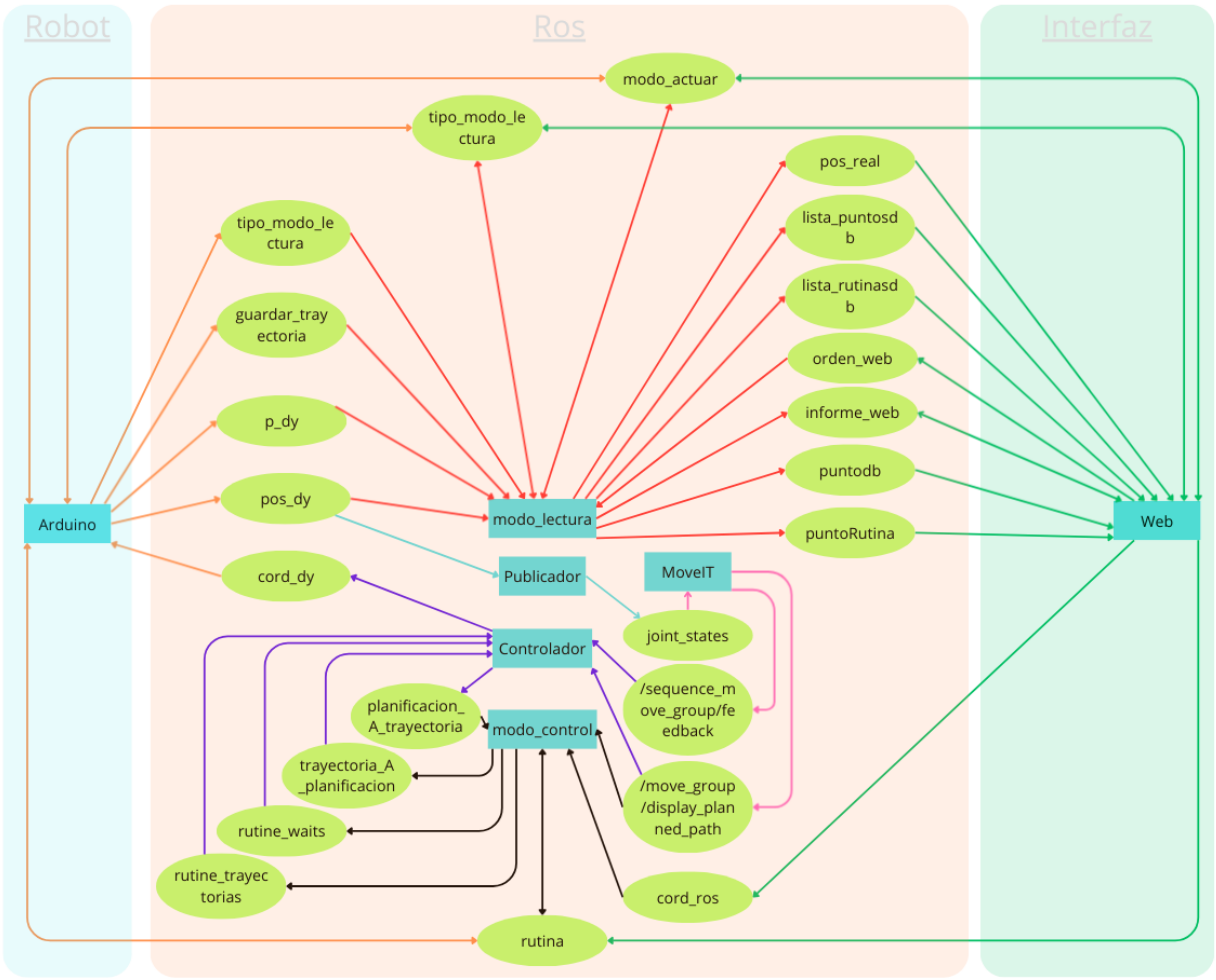
rutine_trayectorias	trayectorias	nombre de las trayectoria y posicion a ejecutar de la rutina	modo_control	controlador_cobot
rutine_waits	waits	tiempo de espera y la posicion en el que se tiene que ejecutar en la rutina	modo_control	controlador_cobot
tipo_modo_lectura	Bool	Indicicador modo actual de modo lectura(Punto-Trayectoria)	Pagina Web arduino modo_lectura	Pagina Web arduino modo_lectura
trayectoria_A_planificacion	Int16	comando para planificacion de ejecuacion rutinas y puntos entre nodos "controlador_cobot" y "modo_control"	modo_control	controlador_cobot

Complementariamente, la misma información se presenta desde la perspectiva de los nodos, organizada por programa o módulo. Esta estructura resulta útil para quienes necesiten modificar o analizar un nodo específico, ya que permite identificar de manera rápida todos los tópicos que utiliza, especificando si cumple el rol de publicador, suscriptor o ambos. [TABLA :]

Programa	Nodo	Topico	USO	Tipo de msg
trayectoria 37	modo_control	rutina	sp	Bool
		cord_ros	s	punto_correr
		planificacion_A_trayectoria	s	Int16
		/move_group/display_planned_path	s	DisplayTrajectory
		trayectoria_A_planificacion	p	Int16
		rutine_waits	p	waits
		rutine_trayectorias	p	trayectorias
Modo lectura 18	modo_lectura	p_dy	s	Int16MultiArray
		orden_web	s	punto_web
		puntodb	p	punto_web
		informe_web	p	String
		pos_dy	s	Int16MultiArray
		/tipo_modo_lectura	s	Bool
		/guardar_trayectoria	s	Bool
		puntoRutina	p	punto_web
		lista_puntosdb	p	nombresPuntos
		lista_rutinasdb	p	nombresPuntos
		pos_real	p	punto_real
		modo_actuar	sp	Bool
publicacion	publicador	pos_dy	s	Int16MultiArray

_posicion_real3	_posicion_rea	joint_states	p	JointState
controlador_rutina_prueba4	controlador_cobot	cord_dy	p	Int16MultiArray
		planificacion_A_trayectoria	p	Int16
		/move_group/display_planned_path	s	DisplayTrajectory
		/sequence_move_group/feedback	s	MoveGroupSequenceActionFeedback
		trayectoria_A_planificacion	s	Int16
		rutine_waits	s	waits
		rutine_trayectorias	s	trayectorias
Pagina Web		cord_ros	p	punto_correr
		orden_web	p	punto_web
		puntodb	s	punto_web
		puntoRutina	s	punto_web
		lista_puntosdb	s	nombresPuntos
		lista_rutinasdb	s	nombresPuntos
		informe_web	sp	String
		pos_real	s	punto_real
		modo_actuar	sp	Bool
		tipo_modo_lectura	sp	Bool
		rutina	p	Bool
arduino	Rosserial	pos_dy	p	Int16MultiArray
		p_dy	p	Int16MultiArray
		rutina	p	Bool
		modo_actuar	sp	Bool
		tipo_modo_lectura	sp	Bool
		cord_dy	s	Int16MultiArray
		guardar_trayectoria	p	Bool

Finalmente, para facilitar aún más la comprensión de la arquitectura del sistema, en la [FIGURA] se incluye un diagrama detallado que representa todas las conexiones entre nodos y tópicos. Este diagrama constituye la vista más completa y visual del intercambio de información dentro del sistema, integrando tanto la parte del robot como la interfaz y los módulos internos de ROS



2.3 Manual de uso

En este capítulo se explicara el funcionamiento y la forma de utilizar el controlador, junto con el robot utilizado para las pruebas; Primero se explicara los diferentes modos que tiene el controlador, luego se presentara cada elemento de la interfaz gráfica y la botonera, por último se explicara cómo utilizar el controlador y como funciona.

2.3.1 Modos de uso

El controlador consta de 2 modos, con 2 sub-modos:

Modo control; en este modo el usuario puede hacer que el robot ejecute las funciones programadas o puntos específicos, a su vez puede ir creando funciones y guardando puntos,

pero solo a través de la interfaz grafica ingresados a través de ella, no guardar los puntos de las posiciones del robot

Modo Lectura; en este modo se bloquea la posibilidad de que el robot ejecute movimientos para protección del usuario, evitando que el robot pueda moverse mientras el usuario esta manipulándolo. Solo en este modo el usuario puede guardar las posiciones del robot real, y desenclavarlo para poder moverlo manualmente.a parte de esto el usuario puede seguir programando rutinas y guardándolas junto con puntos a través de la pagina web. Este modo cuenta con 2 sub-modos: Modo punto y Modo trayectoria.

Modo Punto: En este sub-modo cuando el usuario guarda la posición con [BOTONERA] se guarda un único puto en la rutina con cada pulsación del botón de guardado. Aquí el controlador transforma los valores entregados por el arduino(0 a 4095 y 0 a 1023) a grados y pose, valores que el usuario y ROS respectivamente pueden entender.

Modo Trauectoria: En este sub-modo cuando el usuario guarda la posición con la [BOTONERA] se guarda 20 puntos por segundos mientras el usuario mantenga pulsado el botón de desenclavar, y luego para guardar todos esos puntos en un único archivo el usuario debe presionar el botón de guardado. En este caso se guardan los datos tal cual entran y luego al correrlo se van a entregan a la misma frecuencia sin pasar por el planificador, lo que genera que copie tal cual la trayectoria hasta la velocidad con la que el usuario movió el robot al momento de guardarla.

2.3.2 Elementos

Botonera de control (1.x) [FIGURA:]

1.1-Boton Modo Control B

1.2-Boton Modo Lectura B

1.3-Boton Ejecutar Rutina

1.4-Boton Sub-Modos

1.5-Led Modo Control

1.6-Led Modo Lectura

1.7-Led Ejecutar Rutina

1.8-Led Sub-Modo Punto

1.9-Led Sub-Modo Trayectoria

Botonera de Lectura (2.x) [FIGURA:]

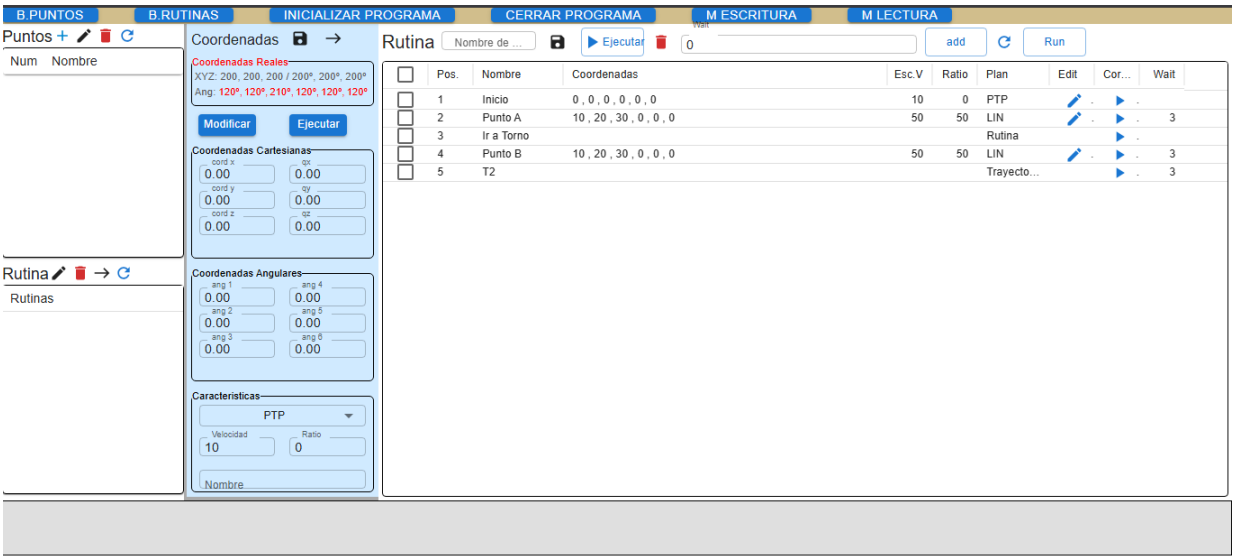
2.1-Boton de Desenclavar

2.2-Boton de Guardado

2.3-Led de Desenclavar

2.4-Led de Guardado

Página Web (3.x)[FIGURA:]



Pantalla de mensajes (3.0) [FIGURA:] :

Barra de herramientas (3.1.x) [FIGURA:] :

3.1.1-Boton de puntos guardados

3.1.2-Boton de Rutinas guardadas

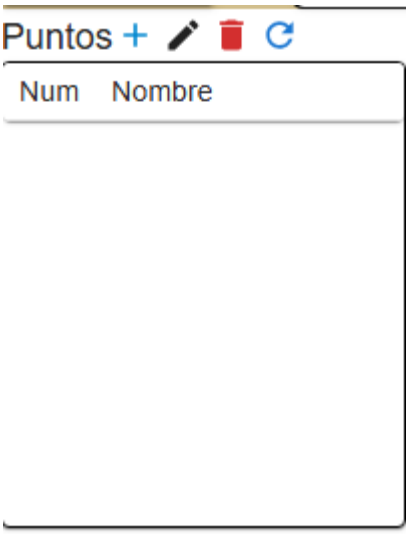
3.1.3-Boton de Inicialización de programa

3.1.4-Boton de Cierre de programa

3.1.5-Boton de Modo Control PW

3.1.6-Boton de Modo Lectura PW

Sección de Rutinas Guardados (3.2.x))[FIGURA:] :



3.2.1-Boton agregar punto guardado a la rutina

3.2.2-Boton de editar Puto

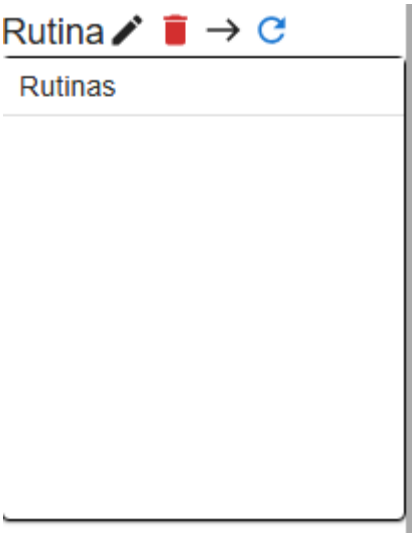
3.2.3-Boton eliminar punto guardado

3.2.4-Boton refrescar lista de puntos guardados

3.2.5-Lista de Puntos Guardados

3.2.6-Punto guardado

Sección de Puntos Guardados (3.3.x) [FIGURA:] :



3.3.1-Boton Editar Rutina

3.3.2-Boton eliminar punto guardado

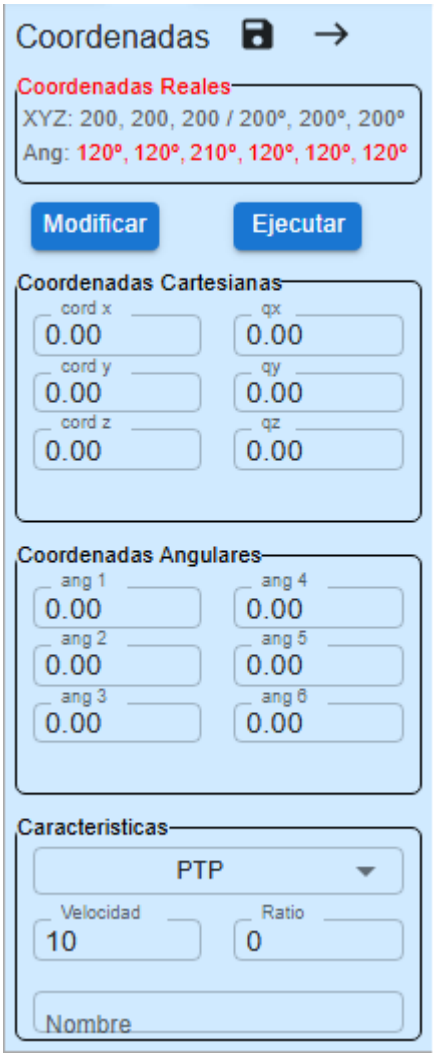
3.3.3-Boton agregar rutina guardada a la rutina

3.3.4-Boton refrescar lista de rutinas guardados

3.3.5-Lista de rutina guardada

3.3.6-Rutina guardada

Sección de Coordenadas (3.4.x) [FIGURA:] :



3.4.1-Boton guardar punto real

3.4.2-Boton agregar punto real a la rutina

3.4.3-Coordenadas reales

3.4.4-Boton guardar punto

3.4.5-Boton agregar punto a la rutina

3.4.6-Inputs coordenadas cartesianas

3.4.7-Inputs coordenadas angulares

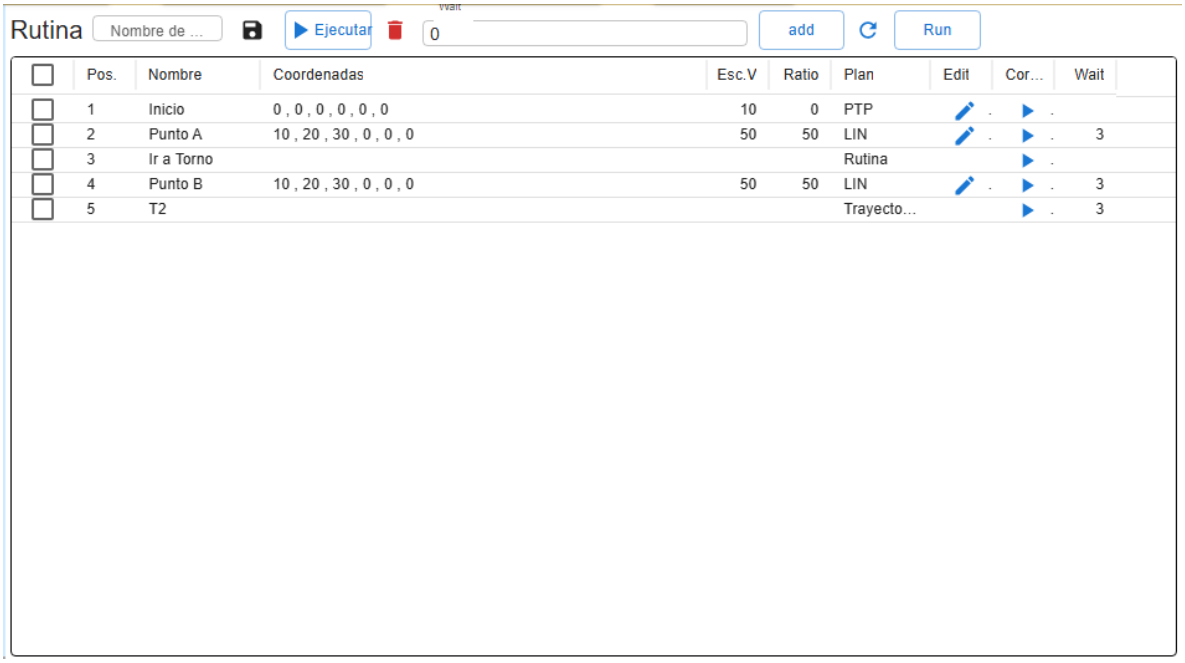
3.4.8-Desplegable tipo de trayectoria

3.4.9-Velocidad del punto

3.4.10-Presicion del punto

3.4.11-Nombre del punto

Sección de Rutina (3.5.x) [FIGURA:] :



3.5.1-Nombre de la rutina

3.5.2-Boton guardar rutina

3.5.3-Boton ejecutar rutina

3.5.4-Boton eliminar puntos seleccionados

3.5.5-Input tiempo de espera

3.5.6-Boton agregar tiempo de espera

3.5.7-Boton refrescar

3.5.8-Cuadro de rutina actual

3.5.9-check de instrucciones de rutina

3.5.10-Posiciones de las instrucciones

3.5.11-Nombres de las instrucciones

3.5.12-Coordenadas de las instrucciones

3.5.13-Velocidad de la instrucción

3.5.1-Presicicion de la instrucción

3.5.1-Tipo de instrucción

3.5.1-Boton de habilitar/confirmar edición de instrucción

3.5.1-Boton para correr instrucción

3.5.1-Tiempo de espera de la instrucción

2.3.3 Descripción:

Primero hay que explicar unos conceptos usados:

Rutina actual (RA): es el conjunto de instrucciones guardada en una tabla volátil de la base de datos, esta es la que se ejecutara cuando se ordene correr la rutina, y donde el usuario programara. Es como un borrador donde se puede probar sin miedo a alterar los datos importantes guardados y sin la necesidad de guardar datos innecesarios al hacer pruebas, esta se borrara automáticamente al iniciar el controlador, en esta se copiaran las rutinas guardadas y solo ingresaran de vuelta a esta parte de la base de datos cuando el usuario realice el proceso de guardado, cuando la rutina creada o editada sea de su agrado. Se visualiza en el **"Cuadro de rutina actual"(3.5.8)**

Tipos de instrucción: En este controlador se tiene 3 tipos de instrucciones que puede hacer que corra el robot:

Punto: El robot va de la posición actual del a las coordenadas indicadas, si se ordena desde la **sección de Coordenadas (3.4.x)** cada motor va al ángulo indicado, en cambio si se ordena desde **sección de Rutina (3.5.x)** el TCP del robot es el que va a la coordenada cartesiana indicada.

Dentro de este tipo de instrucción se puede correr con 2 planes diferentes PTP(punto a punto) donde el robot va a las coordenadas indicadas con el menor movimiento de los motores, y el plan LIN(lineal) donde el robot va al punto indicado con una trayectoria recta

Rutina: El robot realiza de forma ordenada una serie de instrucciones que tiene guardada, si bien por dentro es un conjunto de instrucciones tipo punto, se coloco asi para que el usuario y el controlador pueda diferenciarlas permitiendo que la programación de las rutinas sea modular y no haya que ingresar cada punto que desea ejecutar teniendo ya un conjunto prediseñados de estos.

Trayectoria: Este tipo de instrucción se genera solo en sub-modo trayectoria desde la botonera del robot y representa el conjunto de coordenadas tal cual entrega el arduino, explicado anteriormente en el “modo trayectoria” del “modo lectura”

Comenzaremos explicando cómo funciona la **sección de Rutina (3.5.x)** que es la parte principal del proyecto, en esta se puede visualizar, editar y guardar la (RA), s

“Cuadro de rutina actual”(3.5.8) se visualiza la (RA), en el que cada fila es una instrucción que al momento de correrlas y las columnas características importantes de estas, las cuales se pueden modificar en su mayoría ahí mismo en la tabla

“check de instrucciones de rutina” (3.5.9) sirve para seleccionar varias instrucciones y luego realizar acciones a estas, por el momento solo se pueden eliminar y agregar tiempos de espera.

“Posiciones de las instrucciones”(3.5.10) sirve para indicar el orden con el que se ejecutaran las instrucciones y por dentro como un identificador para algunos procesos

Coordenadas de las instrucciones”(3.5.12): muestra las coordenadas Cartesianas de la instrucción, solo cuando es del tipo punto, para rutina y trayectoria esta casilla esta vacía

Velocidad de la instrucción”(3.5.13): indica la velocidad porcentual a la que se va a mover en esa instrucción siendo la velocidad máxima(100) de 4rad/seg, solo para puntos, para rutina y trayectoria esta casilla esta vacía

Precisión de la instrucción”(3.5.1): indica que tan cerca del punto el TCP va a llegar en esta instrucción en milímetros, esto ayuda a que la trayectoria de la rutina sea mas suave, ejemplo: cuando es 0 el TCP va a ir a la posición especificada, cuando sea 10 va a llegar a 1cm de la

posición especificada y luego seguirá a la siguiente instrucción, solo para puntos, para rutina y trayectoria esta casilla está vacía

Tipo de instrucción”(3.5.1): Indica el tipo de instrucción, esta se puede modificar mediante un menú desplegable solo cuando son del tipo punto y se muestran como PTP o LIN; para rutina y trayectoria no se permite

3.5.1-Boton de habilitar/confirmar edición de instrucción: Normalmente en ninguna casilla se puede ingresar y en la casilla de esta columna aparece un lápiz azul, al apretarlo cambia a un símbolo ✓ y permite que las columnas de esta fila puedan ser modificadas, solo en caso de que sea una instrucción tipo punto, sino solo la columna de tiempo de espera(wait) puede ser modificada esta, para cualquier tipo de instrucción la columna posición no puede ser modificada; esta es la única forma de modificar las instrucciones, sino se deben borrar y volver a ingresarla. En este estado se borra el **“Botón para correr instrucción”(3.5.1)** bloqueando la posibilidad de correr esa instrucción ya que no se puede asegurar que lo mostrado concuerde con lo guardado en la base de datos.

Al volver a apretar sobre el símbolo de esta columna, que ahora es ✓, indica al sistema que quiere guardar la modificación y recién ahí lo envía al controlador, este verifica si es una instrucción valida e informa a la página web, si no es aceptable se muestra el error en la Pantalla de mensajes (3.0) y el símbolo ✓ no cambia, si es aceptable el símbolo de la columna vuelve al del lápiz azul original el controlador guarda la instrucción en la tabla de RA; esto ayuda a que ocurra menos errores al correr rutinas por instrucciones invalidas.

“Botón para correr instrucción”(3.5.1): sirve para que el robot corra esa instrucción, este icono desaparece en “modo lectura” y cuando se está modificando una instrucción.

“Tiempo de espera de la instrucción”(3.5.1): representado como “wait” en la tabla, este es el tiempo en segundos que el robot queda quieto luego de ejecutar la instrucción en donde se está, antes de correr la siguiente instrucción.

“Botón guardar rutina”(3.5.2): se utiliza para guardar la rutina que se encuentra en la RA en la base de datos , lo que pasa internamente es que la base de datos crea una tabla nueva copiando la RA con el nombre ingresado en **“Nombre de la rutina”(3.5.1)**, no es necesario rellenar este input.

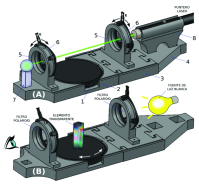
Si no se ingresó ningún nombre el sistema le crea uno genérico que nos e repita con los existentes.

Si el nombre ya existe no se guarda la rutina indicando que esta ya existe

“Botón ejecutar rutina”(3.5.3): Al apretar este botón se enviara la orden al controlador que ejecute la RA, si todo está en orden se le enviara la información al robot para que corra la rutina, sino, si hay cualquier error lo comunica en Pantalla de mensajes (3.0), por ejemplo que no exista alguna de las rutinas que se usen en esta

“Botón eliminar puntos seleccionados”(3.5.4): al cliquear este botón se eliminaran las instrucciones previamente seleccionadas (**“check de instrucciones de rutina” (3.5.9)**) , si alguna instrucción no existe se indicara al usuario por medio de una “ventana emergente”. [FIGURA:].

[FIGURA X]



“Botón agregar tiempo de espera”(3.5.6): al cliquear este botón se agregara a las instrucciones previamente seleccionadas (**“check de instrucciones de rutina” (3.5.9)**) el tiempo indicado ingresado en **“Input tiempo de espera”(3.5.5)**

“Botón refrescar”(3.5.7): al cliquear este botón refresca el **“Cuadro de rutina actual”(3.5.8)** con la información de la RA, se usa por cualquier error que se desincronice la información mostrada con respecto a la que está en la base de datos, el sistema tiene incorporado verificaciones en diversas acciones para detectar alguna desincronización para informarle al usuario asi refresca la tabla.

Seccion 2

En esta sección se puede usar para probar puntos e ingresarlos a la rutina, por aquí se puede comenzar a programar la rutina desde la interfaz web

”():

a veces este cuadro no esta actualizado con la información de la base de datos por algún error (aunque se ha implementado varias confirmaciones para que esto no suceda y si en algún momento igual al ingresar algún dato hay confirmaciones de que la base de datos puede no estar actualizada, informándole al usuario [FIGURA:]) para estos casos se agrego el “Boton refrescar” (3.5.7) que se encarga de re

Tabla Nº1. Abreviaturas

Concepto	Abreviatura
alternating current	ac
analog-to-digital	A–D, A/D
audio frequency*	AF
automatic frequency control*	AFC
automatic gain control*	AGC
amplitude modulation	AM
avalanche photodiode	APD
antireflection*	AR

Fuente: adaptación de [2]

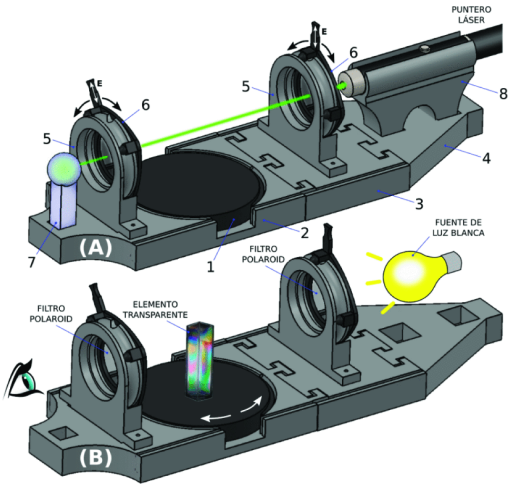


Figura Nº1. Esquema de experiencia sobre polarización.

Fuente: tomado de [4]

CAPITULO X: Análisis de Costos

El Análisis de Costos, conlleva un análisis pormenorizado de los costos involucrados en el proyecto. Puede contar con análisis comparativos, márgenes de rentabilidad, estudios de amortización, etc.

CAPITULO X: Estudio de Impacto Ambiental

El Estudio de Impacto Ambiental, comprende la identificación, evaluación y descripción de los impactos ambientales que producirá el proyecto en su entorno en caso de ser ejecutado, y la descripción de la o las acciones que se ejecutarán para impedir o minimizar sus potenciales efectos adversos. Asimismo, deberá verificarse el cumplimiento de toda la normativa que deba cumplir la temática abordada.

CAPITULO X: Conclusiones

Son las interpretaciones finales que recopilan los datos del trabajo, describe lo que se obtuvo, qué se logró y cuáles son los resultados.

.

Glosario

WIMAX	Técnica de modulación FDM (empleada por el 802.11a y el 802.11g) para transmitir grandes cantidades de datos digitales a través de ondas de radio. OFDM divide la señal de radio en múltiples subseñales más pequeñas que luego serán transmitidas de manera simultánea en diferentes frecuencias al receptor. OFDM reduce la cantidad de ruido (crosstalk) en las transmisiones de señal.
Abstracción	Característica principal de la programación orientada a objetos que se refiere a la capacidad de que un objeto cumpla sus funciones independientemente del contexto en el que se lo utilice; o sea, un objeto “cliente” siempre expondrá sus mismas propiedades y dará los mismos resultados a través de sus eventos, sin importar el ámbito en el cual se lo haya creado.
AES	De las siglas Advanced Encryption Standard o estándar de encriptación avanzada para redes inalámbricas de área local, establecido en la 802.11i, ofrece un nivel de seguridad mayor que el encontrado en el actual estándar de seguridad WPA (Wi-Fi Protected Access).
Agenda Electrónica	Dispositivo con funciones limitadas a anotaciones, contactos, calendario y, en ocasiones, correo electrónico.

Referencias Bibliográficas

[1] “IEEE Reference guie v11.12.2018”. IEEE Periodicals Transactions/Journal Departament, 2018. [En línea]. Disponible en: https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf

[2] “IEEE Editorial Style Manual”, *IEEE Author Center Journals*, 2022. https://journals.ieeeauthorcenter.ieee.org/create-your-ieee-journal-article/create-the-text-of-your-article/ieee-editorial-style-manual/ (consultado el 10 de septiembre de 2022).

[3] R. Hernández Sampieri, C. Fernández Collado, y P. Baptista Lucio, *Metodología de la investigación*, 5a ed. México, D.F: McGraw-Hill, 2010.

[4] R. Peyton, D. Presti, J. H. Martínez Valdiviezo, F. Videla, y G. A. Torchia, “Desarrollo de experiencias para la enseñanza y difusión de la Óptica con impresión 3D [Not available in English]”, en *2020 IEEE Congreso Bienal de Argentina (ARGENCON)*, dic. 2020, pp. 1–6. doi: 10.1109/ARGENCON49523.2020.9505322.

Anexo/s

El documento puede incluir anexos que añadan elementos no esenciales, relativos al tema tratado. Se enumeran sucesivamente. Los anexos contienen información generada por el/la estudiante.

Apéndice/s

Podrán incluirse apéndices que añadan elementos no esenciales, pero que son factibles de considerar en alguna oportunidad, al ser de algún modo relativo al tema tratado. Deben ubicarse luego de los anexos y se enumerarán consecutivamente. Los apéndices contienen información generada por terceros.