Tecnológico de Costa Rica

Escuela de Ingeniería Mecatrónica

Diseño y Validación de la Arquitectura de Control de un Robot Cuadrúpedo

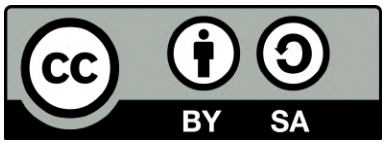
Informe de Proyecto de Graduación para optar por el título de

Ingeniero en Mecatrónica con el grado académico de Licenciatura

José Pablo Vásquez Rojas

Cartago, 11 de marzo 2024

Este trabajo se encuentra bajo una licencia Creative Commons “Atribución-CompartirIgual 4.0 Internacional”.



Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el proyecto realizado y por el contenido del mismo.

José Pablo Vásquez Rojas

Dinamarca, 11 de marzo 2024

Céd: 1-1814-047

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica para ser defendido ante el jurado evaluador, como requisito final para aprobar el curso Proyecto Final de Graduación y optar así por el título de Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: José Pablo Vásquez Rojas

Proyecto: Diseño y Validación de la Arquitectura de Control de un Robot Cuadrúpedo

JUAN CARLOS BRENES TORRES (FIRMA)

Firmado digitalmente por JUAN CARLOS BRENES TORRES (FIRMA)
Fecha: 2024.03.05 11:30:07 -06'00'

MSc. Ing. Juan Carlos Brenes Torres

Asesor

Cartago, 5 de marzo 2024

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: José Pablo Vásquez Rojas

Proyecto: Diseño y Validación de la Arquitectura de Control de un Robot Cuadrúpedo

Digitally signed by FELIPE GERARDO MEZA OBANDO (FIRMA)
Date: 2024.03.15 08:25:35 -06'00'

Miembros del jurado evaluador

CARLOS ADRIAN SALAZAR GARCIA (FIRMA)
Firmado digitalmente por CARLOS ADRIAN SALAZAR GARCIA (FIRMA)
Fecha: 2024.03.15 07:29:29 -06'00'

MSc. -Ing. Felipe Meza Obando	Dr. -Ing. Carlos Adrián Salazar García
Jurado	Jurado

Los miembros de este jurado dan fe de que el presente proyecto final de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

Cartago, 11 de marzo 2024

Resumen

Se presenta el desarrollo de una arquitectura de control para un robot cuadrúpedo en el Laboratorio de Robótica y Control de Aarhus University, Dinamarca, con el objetivo de reducir la discrepancia entre el robot físico y su modelo de simulación. Se identificaron y abordaron deficiencias en la comunicación del robot real, y se diseñó una arquitectura que permite una comunicación eficiente con los actuadores y sensores, alcanzando frecuencias de hasta 700 Hz. Además, se desarrolló una interfaz de pruebas que facilita la adaptación de los algoritmos de control simulados. Asimismo, se implementó la detección de contacto con el suelo y se desarrolló una secuencia de movimiento estable, con errores mínimos de posición y orientación. Estos avances contribuyen significativamente a la mejora de la similitud entre el robot real y su modelo de simulación, impulsando futuras investigaciones en esta plataforma.

Palabras clave: Robótica, Arquitectura de control, Comunicación CAN, Secuencias de caminado.

Abstract

The development of a control architecture for a quadruped robot is presented at the Robotics and Control Laboratory of Aarhus University, Denmark, with the aim of reducing the discrepancy between the physical robot and its simulation model. Deficiencies in the communication of the real robot were identified and addressed, and an architecture enabling efficient communication with actuators was designed, achieving frequencies of up to 700 Hz. Additionally, a test interface was developed to facilitate the adaptation of simulated control algorithms. Furthermore, ground contact detection was implemented, and a stable motion sequence was developed, with minimal errors in position and orientation. These advancements significantly contribute to improving the similarity between the real robot and its simulation model, driving future research on this platform.

Keywords: Robotics, Control architecture, CAN communication, Gaits.

Agradecimientos

Quiero expresar mi profundo agradecimiento a todas las personas que han contribuido de manera significativa a la realización de este proyecto.

En primer lugar, a mi familia, quienes siempre han demostrado un interés genuino en mi bienestar y han brindado su apoyo incondicional a lo largo de mi trayectoria académica. Agradezco su comprensión y paciencia, así como por haberme permitido dedicar el tiempo necesario para completar esta carrera.

Agradezco enormemente a mis profesores y a mi profesor asesor, Juan Carlos, por su constante apoyo y orientación a lo largo de este proyecto y durante los años anteriores. Su experiencia y consejos han sido invaluables en mi desarrollo académico y profesional.

También quiero extender mi gratitud a mis amigos César, Gabriel, y Joshua, quienes compartieron conmigo la pasión por nuestra carrera. A través de su compañerismo y colaboración, convirtieron esta experiencia en algo verdaderamente enriquecedor y memorable.

José Pablo Vásquez Rojas

Dinamarca, 11 de marzo 2024

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

AU	Aarhus University
BW	Ancho de banda
CAN	Controller Area Network
COG	Centro de gravedad
COM	Centro de masa
I²C	Inter-Integrated Circuit Protocol
IMU	Unidad de medición de inercia
IoT	Internet of Things
MPC	Model Predictive Control
OSI	Open Systems Interconnection
ROS	Robot Operating System
RPI	Raspberry Pi
SPI	Serial Peripheral Interface
UART	Universal asynchronous receiver-transmitter
VMC	Virtual Model Control
WBC	Whole-Body Control

Capítulo 1

Introducción

Este proyecto fue desarrollado en el laboratorio de Robótica y Control del Departamento de Ingeniería Mecánica y Producción de Aarhus University, Dinamarca. El mismo se ha dedicado a diversas áreas de investigación en robótica, tales como la integración de robots colaborativos en la industria manufacturera, la inspección de tuberías en pozos petroleros en alta mar, el análisis de dinámica y control de vibraciones en manipuladores robóticos e incluso el análisis de diseño de turbinas eólicas de múltiples rotores [41].

Actualmente se está desarrollando un robot cuadrúpedo que tendrá la capacidad de caminar en diferentes terrenos y a distintas velocidades. Se pretende que este pueda percibir el ambiente y conducir tareas complejas. El robot cuadrúpedo cuenta con una implementación física como la que se muestra en Fig. 1.1, el cual se encuentra en una fase de desarrollo de hardware y desempeño de movimientos controlados.

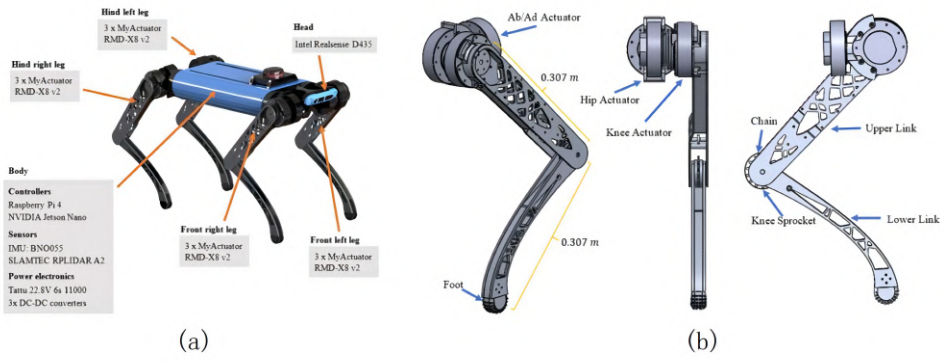


Figura 1.1: Renderizado del robot cuadrúpedo. Recuperado de: [31]

En el laboratorio de investigación se han desarrollado algoritmos de control complejos para la locomoción del mismo, los cuales han sido validados mediante simulaciones. Sin embargo, al querer poner en práctica incluso los más simples, no se ha logrado una implementación exitosa. El esquema de control actual del robot cuadrúpedo mostró ser insuficiente para las necesidades del control de sus articulaciones, mostrando altos tiempos de muestreo al comunicarse con cada uno de sus actuadores. Inicialmente, se contaba

con una única computadora central, correspondiente a una Raspberry Pi 4, la misma estaba conecta a una interfaz de bus CAN, por medio de la cual se controlaban los 12 actuadores. Dicha configuración y esquema de control empleado, ocasionaban que sea imposible actuar sobre cada una de sus articulaciones de manera rápida, lo que resulta en inestabilidades en algoritmos de control que requieren una frecuencia de control relativamente alta.

El laboratorio plantea distintas áreas de investigación con el robot cuadrúpedo que se está desarrollando. Dentro ellas destacan el desarrollo de técnicas de mapeo visual para la navegación autónoma, la integración de dispositivos IoT para establecer interacciones humano-robot y la implementación de algoritmos de inteligencia Artificial para la locomoción adaptativa. A pesar de que cada una de ellas se pueden desarrollar mediante simulaciones, su objetivo final deberá plasmarse físicamente en el robot. El robot presentado es incapaz de ejecutar una locomoción exitosa. Es por ello que resulta imposible poner en marcha proyectos que involucren movimientos coordinados en el robot, la autonomía de su comportamiento o la adaptación a distintos terrenos.

Es decir, la implementación física de su locomoción representa una de las etapas primordiales en la investigación referente al robot y los problemas que se han presentado hasta el momento, constituyen el mayor obstáculo para poner en marcha las demás áreas de investigación del laboratorio. Por ende, este problema merece ser ser abordado detalladamente, siguiendo el proceso de diseño de ingeniería

1.1 Objetivos

1.1.1 Objetivo General

Desarrollar una arquitectura de control que aumente la similitud del robot físico con el modelo de simulación y que permita la implementación de algoritmos de control en diferentes secuencias de caminado

1.1.2 Objetivos Específicos

1. Determinar las principales diferencias entre el robot físico y el modelo de la simulación al nivel de los sistemas mecatrónicos.
2. Desarrollar los cambios necesarios a la arquitectura de control para cerrar la brecha entre simulación y realidad.
3. Implementar el algoritmo de control en el controlador del robot físico para lograr un movimiento adecuado de las patas.
4. Validar el conjunto de mejoras realizadas en la arquitectura e implementación de algoritmo de control mediante al menos una secuencia de caminata en el robot.

1.2 Estructura del Informe

En este trabajo se explican los conceptos más relevantes al diseño de la solución en el capítulo 2. Posteriormente, en el capítulo 3 se brinda una descripción de la metodología empleada para el desarrollo de este proyecto. Con base en dicha metodología, se estructura el desarrollo de la solución en el capítulo 4, en donde se justifica cada una de las decisiones tomadas. En el capítulo 5, se llevan a cabo pruebas en el robot físico con el fin de validar tanto los módulos individuales como el diseño en su totalidad, lo cual se basa en el objetivo específico 4. Luego, se presenta un análisis del impacto económico del proyecto en el capítulo 6, en el cual se abarca los gastos realizados y los beneficios que este le trae al cliente. Por último, se mencionan las conclusiones obtenidas de este proyecto y las recomendaciones en el capítulo 7.

Capítulo 2

Marco Teórico

En este capítulo se presenta los aspectos teóricos más importantes para la elaboración del proyecto. Inicialmente se muestran algunos de los robots cuadrúpedos más populares y sus características más relevantes. Posteriormente, se explican los principales paradigmas de la programación de este tipo de robots, los cuales estructuran el código para lograr un correcto funcionamiento. Luego, se examinan las interfaces de hardware requeridas en la arquitectura de control. Por último, se presentan algunos aspectos generales de la robótica que permiten comprender la manera en que se ejecuta y se interpretan los movimientos de los robots.

2.1 Robots Cuadrúpedos

Los robots móviles con extremidades han sido de un gran interés en los últimos años, gracias a la flexibilidad que poseen para recorrer distintos ambientes. Esto abre las puertas a un gran número de aplicaciones, como lo es en la búsqueda y rescate, la agricultura o la inspección de ambientes industrializados. Una de las configuraciones de este tipo de robots más popular corresponde a los cuadrúpedos, en la cual se alcanza un gran nivel de estabilidad y capacidad para manipular herramientas y transportar objetos. A continuación, se presentan algunos de los robots cuadrúpedos más populares.

2.1.1 MIT Mini Cheetah

Mini Cheetah corresponde a una versión de bajo costo y tamaño reducido de su antecesor, MIT Cheetah 3. Este cuenta con 12 articulaciones accionadas por motores idénticos, correspondientes a motores DC sin escobillas diseñado por ellos mismos. Su diseño mecánico se puede apreciar en Fig. 2.1, en la cual se muestra su implementación física. Su cuerpo versátil y de bajo peso ha permitido que este desempeñe exitosamente algoritmos de control complejos, resultando en velocidades de trote de hasta 2.45 m/s e incluso la ejecución de volteretas hacia atrás [27].

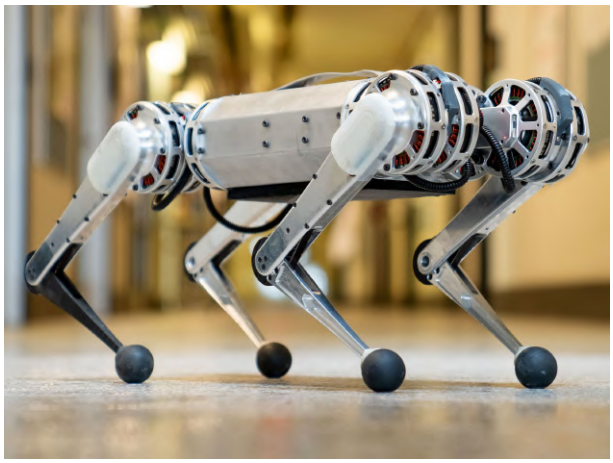


Figura 2.1: Modelo físico del MIT Mini Cheetah. Recuperado de: [9]

2.1.2 ANYmal

ANYmal, mostrado en Fig. 2.3, corresponde al robot cuadrúpedo insignia de la compañía suiza ANYbotics, el cual es autónomo y especializado en el monitoreo de ambientes como la industria de petróleo, de gas, química, minera, entre otras. A diferencia del Mini Cheetah, este posee un mayor peso y tamaño, sin embargo, su diseño está enfocado hacia la locomoción robusta, dinámica y de fácil manipulación y mantenimiento, lo cual es clave para competir en el ámbito industrial. Por otro lado, en este se pueden incorporar múltiples sensores de alta gama y posee un mayor poder de procesamiento, para el cual emplea tres computadoras que se encargan de la locomoción, navegación y la aplicación específica del mismo [23].

Figura 2.2: Modelo físico del robot ANYmal. Recuperado de: [9]

2.1.3 Unitree Go1

El Unitree Go1 es un robot similar al Mini Cheetah en cuanto a especificaciones. Este se enfoca en el mercado industrial, por lo cual, no es de código abierto. Sin embargo, ofrece versiones educacionales en las cuales se puede experimentar aspectos de bajo nivel, incluyendo su locomoción. El mismo posee 3 computadoras, las cuales tienen la capacidad de ejecutar algoritmos de inteligencia artificial de alta complejidad, estas constituyen en conjunto de un CPU de 16 núcleos y un GPU de 1.5 TFLOPS [1]. En comparación con el Mini Cheetah, se puede notar como este posee un poder computacional superior, el cual le permite desarrollar tareas más complejas y de alto nivel empleando sus sensores ópticos y de reconocimiento del entorno. Asimismo, este robot puede alcanzar altas velocidades, de hasta 5 m/s, y soporta una carga adicional de 6kg [1].

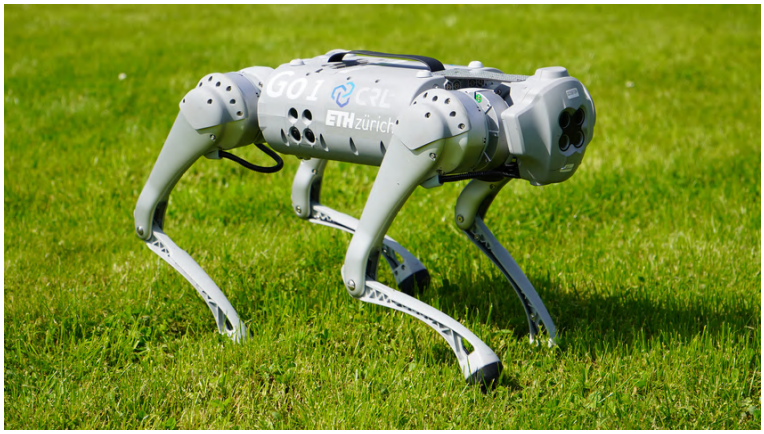


Figura 2.3: Modelo físico del robot Unitree Go1. Recuperado de: [26]

2.2 Paradigmas de programación en robótica móvil

Los robots móviles se pueden desempeñar en un gran número de aplicaciones, sin embargo, ello viene acompañado de una mayor complejidad tanto a nivel de hardware y software. Esto se debe a que están expuestos a una gran variedad de eventos propios del ambiente dinámico. Por lo tanto, sus acciones deben ser ejecutadas con un nivel de lógica que permita una robustez ante estas interacciones y una correcta interpretación de su situación actual para poder completar su objetivo.

El control de los robots móviles se puede estructurar en tres niveles:

- Control de nivel bajo: corresponde a un ciclo de alta frecuencia que controla tareas simples, como lo es determinar la posición a la que se coloca una de las patas del robot [36].
- Control de nivel intermedio: comprende tareas superiores a la locomoción, como lo es la navegación o manipular un objeto [36].

- Control de nivel alto: procesa tareas de alta complejidad con un propósito, como por ejemplo, recoger una herramienta para realizar una acción [36].

En este proyecto se trabajará principalmente en solventar los problemas actuales del control de nivel bajo, ya que la comunicación de los motores y el rendimiento del algoritmo de control son abarcados en este nivel. Algunos de los criterios de diseño con los que debe cumplir este control corresponden a:

- Una comunicación con los actuadores y sensores suficientemente veloz [36].
- Seguridad para evitar consecuencias graves [36].
- El dinamismo para satisfacer cada una de las solicitudes realizadas por los controles de mayor nivel [36].
- La imposibilidad de entrar en un estado de bloqueo [36].

Las tareas que puede ejecutar un robot móvil, desde un punto de vista general, corresponden a sentir, planear y actuar, esto con sus sensores, controladores y actuadores, respectivamente. La manera en que se estructuran estas tareas corresponde a las arquitecturas de control y el llamado paradigma robótico [16]. Existen un gran número de arquitecturas de control, sin embargo, estas se pueden clasificar en 3 grandes grupos.

2.2.1 Arquitecturas Jerárquicas

En este tipo de arquitectura, los robots perciben el entorno y a partir de esta información se crea un mundo estático. Posteriormente se calcula un plan óptimo para cumplir las metas propuestas. Por último, el robot ejecuta las acciones deliberadas, de manera que posteriormente se repite el ciclo hasta cumplir las metas [42]. En Fig. 2.4 se presenta la estructura típica de estas arquitecturas, donde se muestra una serie de capas que define una secuencia de tareas de manera secuencial. Las principales limitaciones de esta arquitectura corresponden a la dificultad de escanear por completo el mapa del ambiente, la incapacidad de responder rápido ante una situación y la gran carga computacional necesaria [42].

2.2.2 Arquitecturas de Comportamiento

En base a las complicaciones experimentadas con la arquitectura jerárquica, nace la necesidad de desarrollar una que sea capaz de adaptarse a un mundo dinámico que cambia constantemente. De esta manera, se desarrolla las arquitecturas de comportamiento o reactivas, las cuales poseen las siguientes capacidades, recuperadas de [42]:

- Reaccionar a ambientes desconocidos y cambiantes.

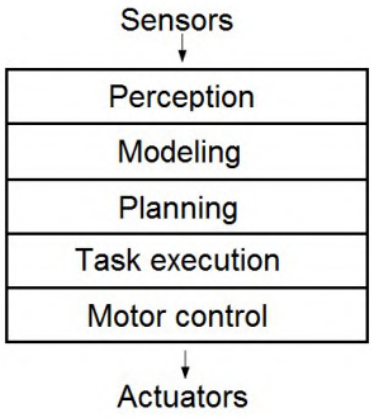


Figura 2.4: Estructura de una arquitectura jerárquica. Recuperado de: [42]

- No se requiere un modelo previo del ambiente o incluso desarrollarlo.
- El tiempo de respuesta al percibir una acción fue disminuido considerablemente.
- Se tiene una mayor robustez, ya que en caso de que una unidad de reacción falle, las demás seguirán funcionando.

En Fig. 2.5 se presenta un ejemplo de una arquitectura reactiva, específicamente una arquitectura de subsunción. En ella se presenta una serie de capas las cuales corresponden a comportamientos que se procesan de acuerdo a la información de los sensores. La ejecución de las acciones se determina por medio de una prioridad, de manera que se logra manejar conflictos y establecer un comportamiento coordinado [42].

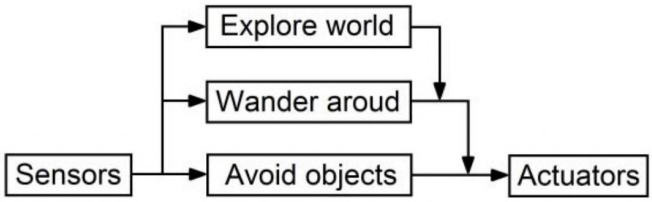


Figura 2.5: Estructura de una arquitectura de comportamiento Recuperado de: [42]

A pesar de que estas arquitecturas de control permiten conllevar el dinamismo del ambiente externo, también se detectaron una serie de desventajas. Entre ellas, se debe mencionar que no son capaces de ejecutar tareas muy complejas debido a la ausencia de la componente de planeamiento, además, es difícil mantener un arbitraje en los comportamientos cuando muchos de ellos quieren ejecutar una acción [42].

2.2.3 Arquitecturas Híbridas

A partir de ello, se plantean las arquitecturas híbridas, las cuales buscan combinar los aspectos positivos de las jerárquicas y las reactivas. En Fig. 2.6 se puede observar la estructura general de la misma, la cual se componen de tres capas:

- Capa deliberativa: proporciona un control de alto nivel para el planeamiento [42].
- Capa reactiva: recibe los comandos de los niveles más altos y se genera una acción [42].
- Capa de ejecución: supervisa la interacción entre las capas de bajo y alto nivel [42].

A partir de ello, se puede inferir que esta arquitectura presenta un comportamiento reactivo en el nivel bajo y deliberativo en el control de nivel alto.

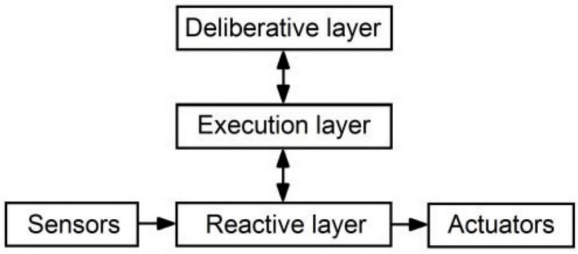


Figura 2.6: Estructura de una arquitectura híbrida Recuperado de: [42]

Este tipo de arquitecturas son flexibles en cuanto a la manera en que se integran los módulos de su estructura, por lo que da paso a un gran número de subtipos. En este caso no existe la presencia de desventajas notables, ya que su propósito es alcanzar un punto óptimo entre las jerárquicas y las reactivas, por lo cual, constituyen un modelo robusto en muchos aspectos, pero que puede ser superado ligeramente en algunos de ellos por los dos tipos de arquitectura abarcados anteriormente.

2.3 Interfaces de Hardware

Las interfaces de hardware permiten adaptar el funcionamiento de un componente físico a la estructura de programación del robot, de manera que se logra un diseño modular en la implementación de los mismos. Una de las plataformas más relevantes para lograr esto corresponde a ROS, el cual fue seleccionado en este trabajo debido a la longevidad que ha presentado a lo largo de los años y el soporte que ha tenido por su comunidad, lo que ha logrado que el mismo pueda considerase un estándar en la programación de robots [29, p. 3].

2.3.1 ROS

ROS, Robot Operating System por sus siglas en inglés, consiste en una serie de herramientas y librerías de código abierto que permiten desarrollar aplicaciones para robots. Este mismo actúa como un "middleware", es decir, corresponde a una capa de software entre el sistema operativo y la aplicación del usuario, la cual tiene como propósito llevar a

cabo tareas en ciertos dominios [29, p. 3]. Además de librerías, ROS posee herramientas de desarrollo y monitoreo, y una metodología de desarrollo.

ROS 2 corresponde a la versión más moderna de este middleware. En este trabajo se decide emplearlo debido al soporte que tendrá en un futuro, además de que incluye algunas características que pueden ser de utilidad en la programación de bajo nivel, como lo es el funcionamiento en tiempo real, el control de calidad en la comunicación y la asincronía en la ejecución de servicios [35].

Generalidades de ROS2

Las distribuciones de ROS 2 poseen la máxima compatibilidad con el sistema operativo Ubuntu - Linux, sin embargo, también se pueden instalar en otros sistemas operativos, pero el correcto funcionamiento no está asegurado porque se consideran plataformas experimentales. ROS 2 emplea en gran parte la programación orientada a objetos, en el lenguaje `C++` o Python. A continuación, se presentan algunos elementos relevantes en la estructura de ROS, recuperados de [29, p. 8].

- **Nodo:** es un objeto en la red de ROS, el cual puede comunicarse con otros.
- **Tema:** corresponde a un canal de comunicación por el cual se envían mensajes de un tipo específico.
- **Publicación/Subscripción:** consiste en una comunicación asíncrona en donde los nodos publican mensajes a un tema.
- **Servicios:** es una comunicación asíncrona en donde un nodo realiza una solicitud a otro nodo y este responde inmediatamente.
- **Acciones:** también corresponde a una comunicación asíncrona, en la cual un nodo realiza una solicitud a otro, sin embargo, esta misma tomará un tiempo en completarse, por lo que el nodo solicitante recibirá realimentación del avance de la acción y cuándo esta alcanza la meta deseada o en caso contrario, si hubo un error en el proceso.

2.4 Protocolos de Comunicación

En este proyecto se emplean múltiples componentes físicos, los cuales se comunican con la computadora central de manera distinta. En esta sección, se abarcan los protocolos más relevantes en el diseño del proyecto, debido a que forman parte de la toma de decisiones del proceso de diseño.

2.4.1 CAN

El protocolo de comunicación CAN, Controller Area Network por sus siglas en inglés, es utilizado ampliamente en el sector industrial debido a su alto desempeño y la flexibilidad para realizar modificaciones en la red. El modelo OSI de 7 capas se presenta en Fig. 2.7. Las dos capas inferiores de este modelo están definidas por el estándar ISO-11898:2003 [12, p. 2]. Estas capas comprenden un controlador encargado de gestionar los paquetes de datos y generar comandos. Posteriormente, un transceptor emite la señal hacia un bus que se compone de un par de cables eléctricos [12, p. 2].

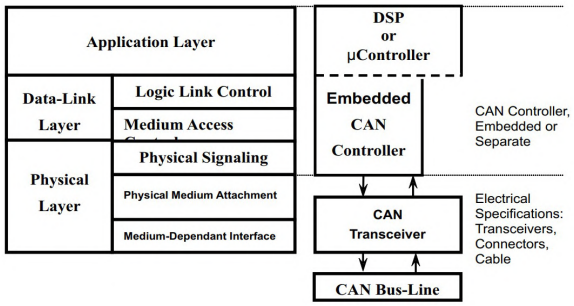


Figura 2.7: Capas del protocolo de comunicación CAN. Recuperado de: [12, p. 2].

Se destacan las siguientes características del protocolo, recuperadas de [12, p. 3]:

- Carrier Sense: los nodos de la red escuchan al bus antes de transmitir para verificar su disponibilidad .
- Multiple Access: diversos nodos tienen acceso al bus y pueden transmitir datos.
- Detección de Colisiones: las colisiones se resuelven de acuerdo con la prioridad.
- Prioridad de mensajes: el acceso al bus es controlado mediante una prioridad ligada al identificador del nodo.

La trama de datos de este protocolo se puede observar en Fig. 2.8, la cual está conformada por los siguientes elementos, recuperados de [12, 3]:

- SOF (1 bit): marca el inicio del mensaje.
- Identificador (11 bits): establece la prioridad del mensaje; cuanto más bajo sea este valor, mayor será la prioridad. También identifica al nodo dentro de la red.
- RTR (1 bit): notifica si otro nodo requiere la información.
- IDE-A (1 bit): indica si la trama de datos es de tipo estándar.
- r0 (1 bit): reservado para futuras modificaciones del estándar.

- DLC (4 bits): indica el número de bytes transmitidos.
- Datos (hasta 64 bits): corresponde a los datos transmitidos desde la aplicación.
- CRC (16 bits): proporciona redundancia para la detección de errores.
- ACK (2 bits): confirma la recepción e integridad del mensaje.
- EOF (7 bits): indica el final de la trama.
- IFS (7 bits): contiene una etiqueta de tiempo que permite al controlador colocar el mensaje en la posición adecuada dentro del buffer de mensajes.

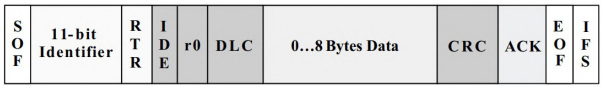


Figura 2.8: Trama de datos estándar del protocolo CAN. Recuperado de: [12, p. 3].

La trama de datos estándar es usada en el protocolo CAN 2.0A, mientras que es posible extender su identificar a 29 bits en lugar de 11 bits, lo cual corresponde al protocolo CAN 2.0B [11]. Al transmitir mensajes, este protocolo se distingue por su enfoque en el contenido del mensaje en lugar de la dirección de destino, a diferencia de otros protocolos. Por otro lado, su manejo de fallos le permite que un nodo no funcional no deje inservible una red, ya que este es expulsado automáticamente del bus [11]. Esto resulta especialmente útil en sistemas con múltiples actuadores, ya que el fallo en la comunicación con un actuador no pone en riesgo la estabilidad de los otros.

2.4.2 I^2C

El protocolo I^2C se emplea en la comunicación entre la computadora central y sensor de inercia (IMU). Este tipo de comunicación se caracteriza por la presencia de un maestro y uno o varios esclavos. Esta corresponde a una interfaz bidireccional, por lo cual los esclavos pueden enviar mensajes al controlador en caso de que se les solicite. El enrutamiento de los mensajes se realiza por medio de una dirección única presente en cada esclavo [38]. Sus conexiones consisten de 2 cables de procesamiento, SDA y SCL, y 2 para suplir energía al componente esclavo en caso de ser necesario, tal como se muestra en Fig. 2.9.

En dicho diagrama también se observa dos resistencias pull-up en los canales de comunicación, las cuales son necesarias en caso de que el módulo no las posea. Sin embargo, el sensor de inercia usado actualmente, correspondiente al BNO055 Xplained Pro, no necesita de ello. En esta figura se observa que la fuente de poder VCC no alimenta el módulo esclavo, ya que es únicamente referente a la comunicación, pero en la mayoría de los casos, dichos componentes se pueden suplir con la terminal de la computadora central, tal como es el caso con el IMU empleado.

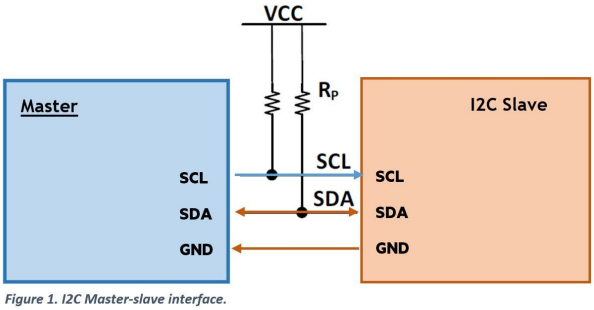


Figura 2.9: Conexiones del protocolo I^2C. Recuperado de: [34]

2.4.3 USB

Si bien el protocolo de comunicación USB no es un aspecto de análisis profundo en este trabajo, resulta conveniente conocer sus características para justificar la toma de decisiones realizada. Para el caso de USB 2.0, este permite tasas de transferencia de hasta 480 Mbps y hasta 127 dispositivos físicos en la red. Además, admite paquetes de diferentes tamaños, un manejo del flujo para buffers y manejo de errores [10].

Para el caso de USB 3.0, este posee una arquitectura de bus dual, lo cual le permite ser compatible con USB 2.0. Por otro lado, esta versión es capaz de alcanzar tasas de transferencias de 5.0 Gbps, lo cual es muy superior a las de su predecesora [21].

2.5 Fundamentos de Robótica

En esta sección se presentan los términos más importantes de la descripción de los movimientos y fuerzas a los que se someten los brazos robóticos.

A pesar de que en este trabajo se emplea un robot que posee piernas, el concepto de brazo robótico se puede aplicar individualmente a cada una de ellas. Un brazo robótico está compuesto de una serie vínculos, los cuales conectados mediante articulaciones que permiten el movimiento de los mismos [13, p. 5]. Existen distintos tipos de articulaciones, sin embargo, las giratorios o angulares corresponden a las relevantes en este trabajo.

2.5.1 Descripción espacial

La descripción del posicionamiento y orientación de los elementos del robot se realiza mediante sistemas de referencia, comúnmente representados en coordenadas cartesianas tridimensionales. Al asignar coordenadas a partes específicas del manipulador, se facilita la interpretación matemática de sus movimientos, y a esta asignación de coordenadas se le conoce como trama.

En este contexto, se logra representar la posición y orientación de un objeto, como se ilustra en Fig. 2.10. La figura muestra la ubicación de un efector final, representado por

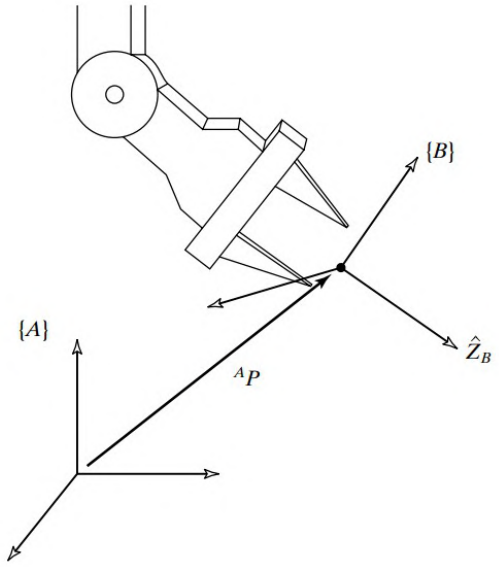


Figura 2.10: Representación de la ubicación y orientación de un objeto. Recuperado de: [13, p. 21]

la trama {B} y que emplea la trama {A} como referencia. En ella, la posición se expresa mediante el vector ^AP. Sin embargo, para describir la orientación, es necesario recurrir a la orientación propia de la trama B. Para ello, se utiliza una transformación de rotación, que detalla cómo los tres ejes de las coordenadas de {B} fueron rotados, teniendo a {A} como punto de origen, como se expone en la ecuación 2.1, de [13, p. 21].

$$ {}^A_BR = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.1} $$

De esta manera, se entiende que las coordenadas del efector corresponden a una traslación equivalente a $^AP_{BORG}$ y una posterior rotación ^A_BR. A partir de ello, para representar un punto en el espacio de la trama B en términos de la trama A, se emplea una transformada homogénea, en este caso denominada ^A_BT, la cual consiste en una matriz 4x4 que unifica las dos transformaciones mencionadas anteriormente, la misma se presenta en las ecuaciones 2.3 y 2.2, de [13, p. 28].

$$ {}^AP = {}^A_BT\,{}^BP \tag{2.2} $$

$$ {}^A_BT = \begin{bmatrix} {}^A_BR & {}^AP_{BORG} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{2.3} $$

2.5.2 Cinemática Directa

Estas transformaciones entre tramas se pueden propagar, de manera que, al asignarle coordenadas a múltiples elementos del robot, es posible describir sus movimientos en términos de una trama base, mediante la matriz de transformación adecuada. Se deben tomar en cuenta todos los elementos entre el objeto de interés y la base, debido a que, al estar unidos en una cadena cinemática, su posición, velocidad y aceleración depende de cada uno de ellos. El objeto de mayor interés en el brazo robótico corresponde al efector final, pues es el que deberá interactuar con el entorno. Es por ello que el principal objetivo de la descripción cinemática del manipulador corresponde a encontrar la matriz de transformación que permita representar este objeto en términos de una trama base, y que dependa de las variables de las demás articulaciones.

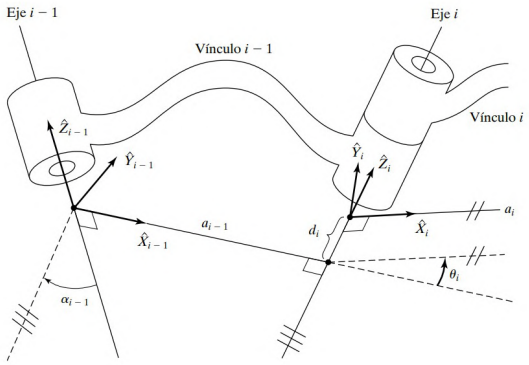


Figura 2.11: Relación entre las tramas de las articulaciones. Recuperado de: [13, p. 66]

Para lograr esto, se emplea un procedimiento que consiste en encontrar los parámetros de Denavit-Hartenberg, los cuales establecen variables de interés al relacionar dos articulaciones unidas por un vínculo rígido, tal como se aprecia en la figura 2.11, en donde se recalcan los parámetros:

- a_i: corresponde la distancia entre los dos ejes $\hat{Z}_i$ y $\hat{Z}_{i+1}$ medido sobre el eje $\hat{X}_i$ [13, p. 69].
- α_i: es el ángulo entre los dos ejes $\hat{Z}_i$ y $\hat{Z}_{i+1}$ medido sobre $\hat{X}_i$ [13, p. 69].
- d_i: corresponde la distancia entre $\hat{X}_{i-1}$ y $\hat{X}_i$ medido sobre el eje $\hat{Z}_i$ [13, p. 69].
- θ_i: es el ángulo entre $\hat{X}_{i-1}$ y $\hat{X}_i$ medido sobre $\hat{Z}_i$ [13, p. 69].

Es importante mencionar que el posicionamiento de las coordenadas sobre ambos ejes de las articulaciones debe seguir una serie de reglas, sin embargo, no se pretende llegar a ese nivel de profundidad en este trabajo.

Con base en estos parámetros, se desarrolla la matriz de transformación entre dos tramas consecutivas, la cual se puede observar en la ecuación 2.4, recuperada de [13, p. 75].

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_1 & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4} $$

Este proceso se realiza para cada una de las articulaciones, de manera que se puede llegar finalmente a representar la posición del efector final, el cual posee la trama {N}, en términos de la base {0}, tal como se presenta en la ecuación 2.5, de [13, p. 76].

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,\ldots\,{}^{N-1}_{N}T \tag{2.5} $$

En el caso de un brazo robótico que posee únicamente articulaciones giratorias, se puede representar las coordenadas $\begin{bmatrix} {}^{N}\hat{X}_0 & {}^{N}\hat{Y}_0 & {}^{N}\hat{Z}_0 \end{bmatrix}$ actuales del efector final, al evaluar la matriz ${}^{0}_{N}T$ con los parámetros previamente definidos a_i, α_i y d_i y al leer los ángulos actuales de cada una de las articulaciones, correspondientes a θ_0, θ_1, ... θ_N.

2.5.3 Cinemática Inversa

La cinemática directa permite obtener la ubicación actual de un elemento del robot al leer las variables de las articulaciones, sin embargo, esta no resulta directamente útil para comunicarle al robot que se mueva hacia una posición específica en el espacio cartesiano. Para solventar dicho problema se emplea la cinemática inversa, la misma consiste en realizar un análisis de la geometría del robot para encontrar un conjunto de ángulos de las articulaciones θ_0, θ_1, ... θ_N, que le permita al robot llegar a un punto cartesiano de interés. La solución también se puede plantear de una manera iterativa en lugar de analítica, sin embargo, la complejidad del cálculo aumenta significativamente [13, p. 6].

La existencia de soluciones a dicho problema matemático depende estrictamente del alcance del robot, es decir, su espacio de trabajo. También existe la posibilidad que se obtengan múltiples soluciones, de manera que se debe definir previamente el conjunto de posibles ángulos a los que se le comandar el robot [13, p. 102]. Existen dos tipos de enfoques para resolver la cinemática inversa desde un punto analítico, los cuales corresponden a el algebraico y geométrico. A continuación, se mencionará el funcionamiento de este último, ya que la solución que se ha planteado para el robot en cuestión fue realizada de esta manera.

El método geométrico resulta útil cuando las articulaciones poseen rotaciones α_i entre sus ejes de 0° o ±90° [13, p. 112]. En base a esto, se analiza visualmente la geometría de sus vínculos. Tómese por ejemplo el brazo robótico de Fig. 2.12, el cual tienen 3 articulaciones y la totalidad de ellas presentan sus ejes $\hat{Z}$ alineados. A partir de ello se determina la expresión de θ_1, θ_2 y θ_3 en base al cálculo de los ángulos β y Ψ. Para la determinación de los ángulos se realizan operaciones aritméticas como trigonometría y ley de cosenos.

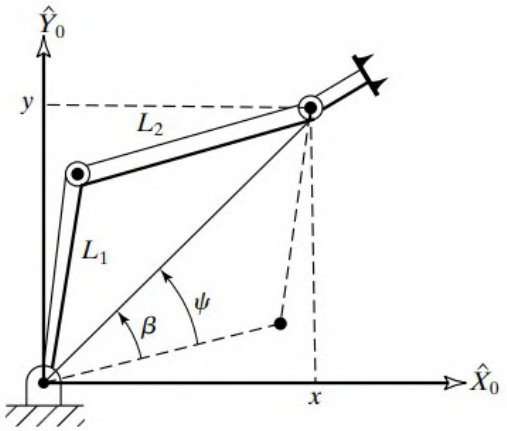


Figura 2.12: Ejemplo de la aplicación del método geométrico para el desarrollo de la cinemática inversa de un brazo robótico de 3 articulaciones. Recuperado de: [13, p. 112]

2.5.4 Estática de Manipuladores

Los procedimientos anteriores se centran en describir al brazo robótico en términos de las posiciones. Sin embargo, el robot también tendrá propiedades físicas y una interacción con el ambiente, por lo cual, es de vital importancia entender las fuerzas estáticas a los que se somete y la manera en que la velocidad de los elementos de su cadena resulta relevante.

Si bien el desarrollo de las ecuaciones que describen la velocidad angular de las articulaciones son importantes en el cálculo de las fuerzas, no se entrará en detalle debido a que las mismas han sido elaboradas y validadas en trabajos anteriores en el robot cuadrúpedo. Sin embargo, sí vale la pena indagar en el término de los jacobianos, ya que constituyen la manera en que dichas ecuaciones resultan de gran utilidad en la vida real.

El jacobiano hace referencia a una manera de expresar una derivada multidimensional. En la ecuación 2.6, recuperada de [13, p. 149] se observa el término Y, el cual consiste en una matriz que alberga los n resultados de una cantidad n de funciones, F, la cual posee n variables independientes, X.

$$Y = F(X) \tag{2.6}$$

Al calcular los diferenciales de Y en función de los diferenciales de X, se obtiene la ecuación 2.7, esto mediante la regla de la cadena.

$$\partial Y = \frac{\partial F}{\partial X} \partial X \tag{2.7}$$

La matriz de derivadas parciales $\frac{\partial F}{\partial X}$ es lo que se conoce como jacobiano, $J(\Theta)$. Al aplicar la diferenciación por el tiempo se observa la relación de las velocidades, en la ecuación 2.8.

$$\dot{Y} = J(X)\,\dot{X} \tag{2.8}$$

En el área de la robótica resulta especialmente útil la relación entre las velocidades cartesianas del efector final con las velocidades angulares de las articulaciones, tal como se observa en la ecuación 2.9, recuperada de [13, p. 150].

$${}^{0}v = \begin{bmatrix} {}^{0}v \\ {}^{0}\omega \end{bmatrix} = {}^{0}J(\Theta)\,\dot{\Theta} \tag{2.9}$$

La utilidad de los jacobianos no se extiende únicamente a las velocidades. A partir de la ecuación 2.9 y al emplear la definición de trabajo, que constituye de una fuerza aplicada a través de una distancia, se puede encontrar la relación de los torques de las articulaciones en función de la fuerza aplicada al efector final, tal como se observa en la ecuación 2.10 [13, p. 157].

$$\tau = J^{\intercal}F \tag{2.10}$$

A partir de ello, se pueden desarrollar controladores que calculan las fuerzas de reacción necesarias, y mediante la ecuación presentada anteriormente, permiten enviar los comandos necesarios de torque para que el brazo se mantenga en una posición estática.

2.6 Entornos de Simulación

El desarrollo de algoritmos de control en robot cuadrúpedos es un proceso complejo, dada la gran cantidad de variables a tomar en cuenta y las consideraciones físicas a la hora de implementarse en el robot real. Es por ello que es preferible llevar a cabo dicho proceso en una herramienta de simulación, la cual permita modificar fácilmente el diseño actual y a su vez, poder observar su rendimiento. Este trabajo se enfocará en la herramienta Webots, ya que es la que se ha empleado en el robot cuadrúpedo en los trabajos más recientes.

2.6.1 Webots

Webots es un software de código abierto utilizado para simular robots en diversos sistemas operativos, siendo ampliamente aplicado en la industria e investigación. En esta plataforma, es posible integrar una variedad de componentes, como sensores, actuadores, cuerpos rígidos, y diferentes materiales, con el propósito de desarrollar y validar algoritmos de control. Los controladores resultantes pueden estar escritos en varios lenguajes de programación, como C, C++, Python, Java, MATLAB o ROS [14].

En Fig. 2.13 se muestra las posibilidades del software, en donde existe un ambiente interactivo con valida el rendimiento del control de un dron, además de incluso capturar otros datos como la imagen que percibe la cámara del mismo.

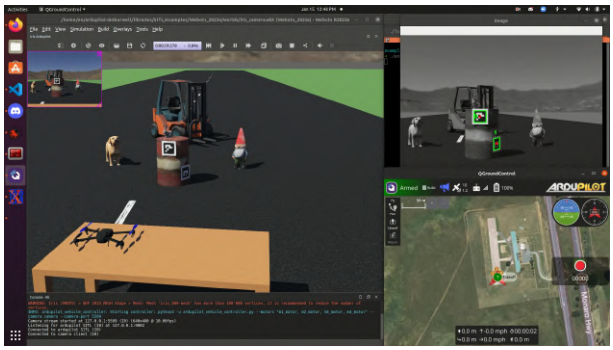


Figura 2.13: Ejemplo de una simulación en el software Webots. Recuperado de: [3]

2.7 Trabajos Anteriores

El robot cuadrúpedo con el que se trabajó fue desarrollado completamente en el Laboratorio de Robótica y Control de Aarhus University. Este inició como un proyecto de desarrollo e investigación de dos estudiantes de maestría, Simon Jeppesen y Søren Ingwersen, en donde se abarcó la selección de los actuadores, los sensores, los módulos de procesamiento y el dimensionamiento de los demás componentes eléctricos, además del diseño mecánico de sus vínculos [30]. En Fig. 2.14 se puede observar el modelo físico preliminar del robot cuadrúpedo, en donde se puede destacar el uso de piernas impresas en 3D en lugar de aluminio, y la falta de una carcasa, los cuales fueron implementados posteriormente.

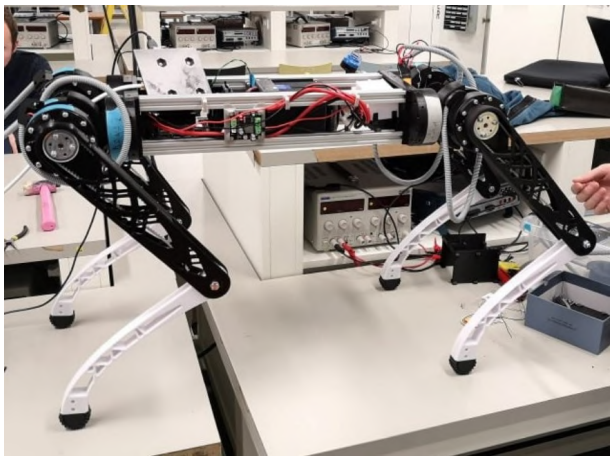


Figura 2.14: Construcción preliminar el robot cuadrúpedo. Recuperado de: [30]

2.7.1 Towards a Kinematic and Dynamic Control System for a Four-Legged Robot

Posteriormente, ambos estudiantes realizaron su tesis de maestría, la cual buscaba avanzar hacia un control cinemático y dinámico del robot cuadrúpedo [31]. En ella se desarrollaron los cálculos de la cinemática directa e inversa de las piernas del robot. Si bien los resultados de la cinemática directa estaban correctos en su momento, modificaciones posteriores al modelo físico les restó su utilidad, por lo cual se presentarán más adelante. Sin embargo, cabe recalcar el cálculo de la cinemática inversa, el cual fue especialmente útil en este trabajo.

En Fig. 2.15 se observa las coordenadas de referencia usadas en dicho cálculo. Para reducir el tamaño de las ecuaciones, se presentan dos magnitudes de interés, $\overline{AG}$ $\overline{AC}$, las cuales se pueden obtener mediante las ecuaciones 2.11 y 2.12, respectivamente.

$$AG = \sqrt{x^2 + y^2 - {L_1}^2} \tag{2.11}$$

$$AC = \sqrt{AG^2 + z^2} \tag{2.12}$$

Para calcular los ángulos de las 3 articulaciones cada una de las piernas, θ_1, θ_2 y θ_3 , se emplean las ecuaciones 2.13, 2.14 y 2.15, respectivamente.

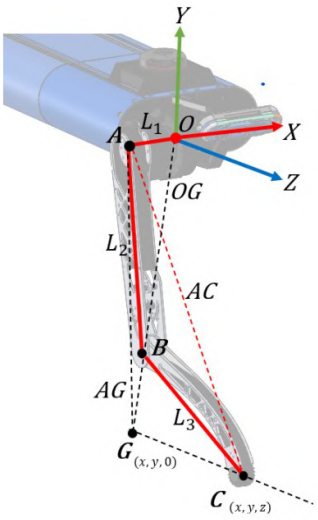


Figura 2.15: Coordenadas de referencia para el cálculo de la cinemática inversa en una de las piernas del robot. Recuperado de: [31]

$$\theta_1 = -\arctan 2(AG, OA) - \arctan 2(y, x) \tag{2.13}$$

$$\theta_2 = \arctan 2(z, AG) - \arccos\left(\frac{AC^2}{2 \cdot L_3 \cdot AC}\right) \tag{2.14}$$

$$\theta_3 = \pi - \arccos\left(-\frac{AC^2 - L_2^2 - L_3^2}{2 \cdot L_2 \cdot L_3}\right) \tag{2.15}$$

En dicho trabajo también se realiza un modelado dinámico del robot, sin embargo, el mismo no ha sido utilizado hasta la fecha. Además, se desarrolló dos secuencias de caminado, el gateo y el trote. En la secuencia de gateo se planteó mover lateralmente el cuerpo del robot en un inicio, para luego balancear una de las piernas mientras las otras 3 empujan el cuerpo hacia adelante, tal como se observa en Fig. 2.16. Por otro lado, la secuencia de trote implica balancear dos de las piernas mientras las otras dos mueven el cuerpo hacia adelante y tocan el suelo, como se muestra en la Figura 2.17. Ambas secuencias de caminata fueron verificadas en el entorno de simulación de Gazebo.

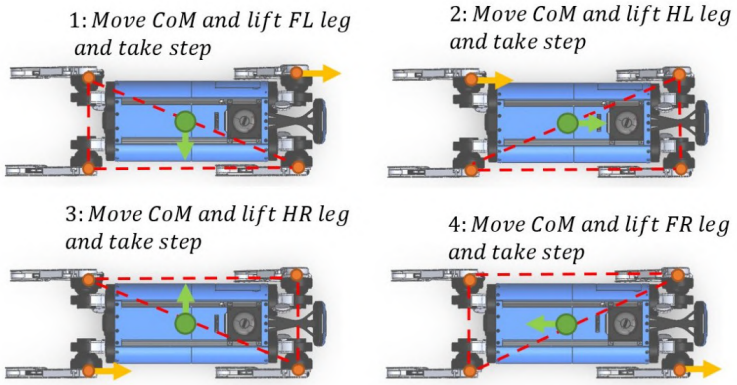


Figura 2.16: Secuencia de gateo diseñada previamente. Recuperado de: [31]

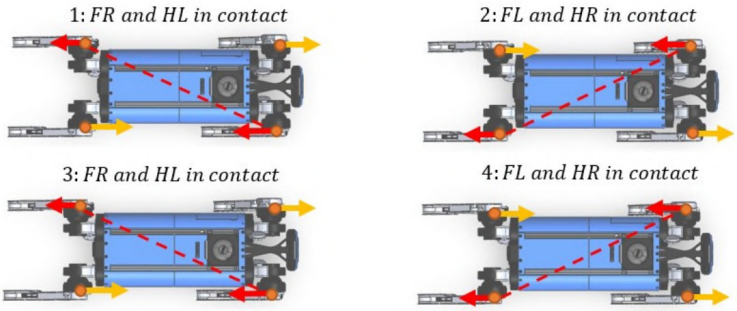


Figura 2.17: Secuencia de trote diseñada previamente. Recuperado de: [31]

Por último, en este trabajo se abarcó ligeramente el diseño de la arquitectura de control del robot. Esta se presenta en Fig. 2.18, en la cual se observa una serie de procesos y cálculos secuenciales, es decir, no existe ninguna prioridad en alguno de ellos y el proceso más lento será el cuello de botella en el rendimiento del mismo. De esta manera, una interacción con el mundo real que afecte a los sensores deberá recorrer todos los subprocesos de la arquitectura para que por fin se realice una acción en base a ello. Es decir, se determina que dicha arquitectura no posee un comportamiento reactivo.

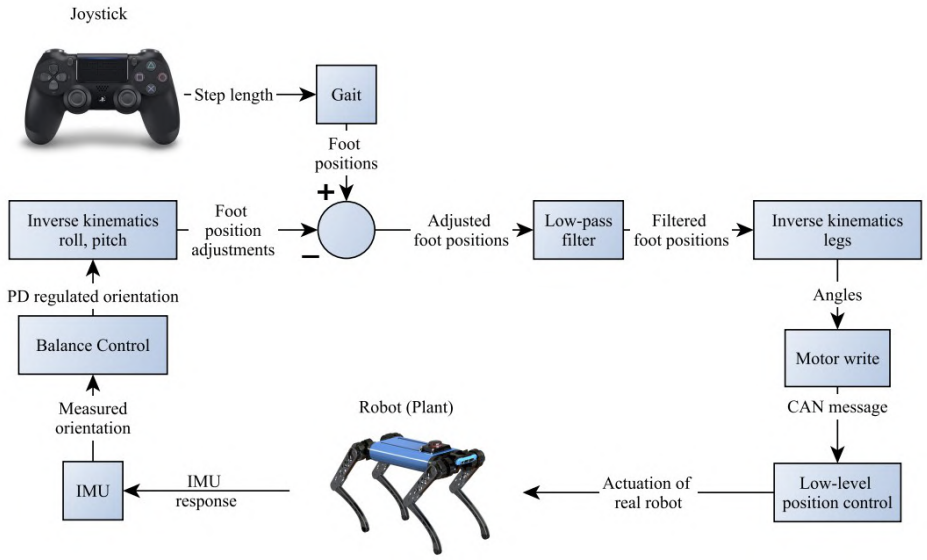


Figura 2.18: Arquitectura de control diseñada previamente. Recuperado de: [31]

2.7.2 Locomotion Control Scheme for Quadruped Robots

En un trabajo posterior, el estudiante de maestría Yuxiong Du, desarrolló dos algoritmos de control para la locomoción del robot cuadrúpedo [15]. El primero de ellos corresponde a VMC, Virtual Model Control por sus siglas en inglés. Este consiste en usar elementos virtuales como resortes y amortiguadores para establecer un comportamiento del robot con el ambiente, posteriormente se generan las fuerzas virtuales necesarias para alcanzar el movimiento deseado. En Fig. 2.19 se puede observar el funcionamiento del algoritmo VMC en 2 dimensiones al balancear una de las piernas, en donde las dos fuerzas virtuales, f_x y f_y, están definidas por un comportamiento de amortiguador y un resorte.

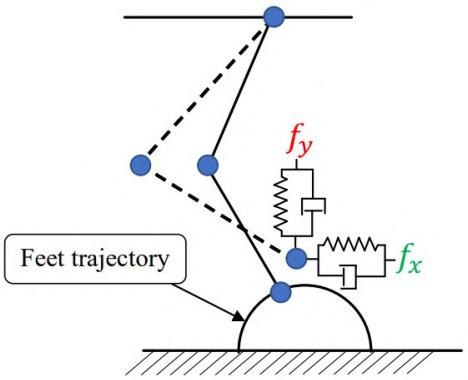


Figura 2.19: Algoritmo de control VMC en el balanceo de una pierna. Recuperado de: [15]

El cálculo de estas fuerzas virtuales se puede ver en la ecuación 2.16, en la cual se presentan dos coeficientes k_u y c_u, los mismos constituyen los coeficientes de rigidez y amortiguamiento, respectivamente.

$$f_u = k_u(u_d - u) + c_u(\dot{u}_d - \dot{u}) \tag{2.16}$$

Por otro lado, u y $\dot{u}$, constituyen la posición y velocidad del efector final y el subíndice d hace referencia al valor deseado. Este control resulta efectivo cuando se desea seguir una trayectoria, debido a que se necesita tomar en cuenta tanto la posición como la velocidad del efector final [15].

El otro algoritmo de control que se desarrolló en dicho trabajo fue el MPC, Model Predictive Control por sus siglas en inglés. Su funcionamiento se basa en predecir el comportamiento del sistema en un futuro, para luego ajustar las entradas de control y poder alcanzar una trayectoria deseada. Este resulta menos intuitivo que VMC, pero su desempeño en término de robustez a perturbaciones es mucho mejor. En Fig. 2.20 se puede observar la mejora en el rendimiento que conlleva este algoritmo, tanto en la reducción del error como en una mayor estabilidad, esto al examinar los cambios de la orientación del centro de masa en Fig. 2.20.b y Fig. 2.20.c.

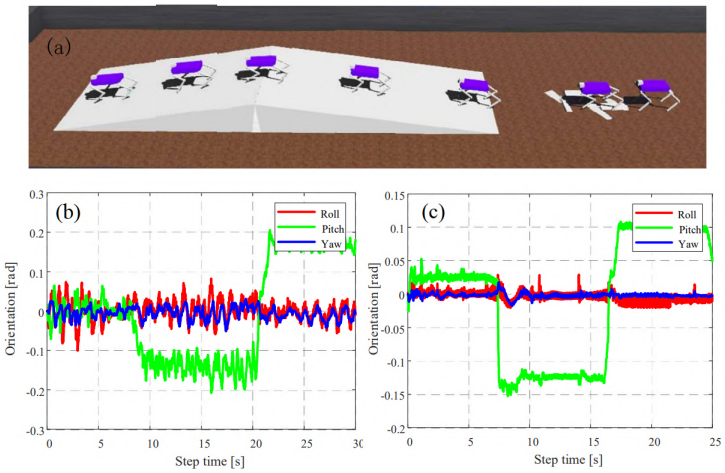


Figura 2.20: Resultados del desempeño de los algoritmos de control. (a) Trayecto de la simulación, (b) orientación del cuerpo del robot empleando el control VMC, (c) orientación del cuerpo del robot usando el control MPC. Recuperado de: [15]

2.7.3 Model-Based Control Strategy for a Quadruped Robot

Posteriormente, el estudiante de maestría Lixuepiao Wan trabajó en un proyecto de investigación y desarrollo en donde se buscaba continuar el desarrollo de los controladores VMC y MPC [40], con el fin de adaptarlos al robot en estudio y disminuir las simplificaciones realizadas en los trabajos anteriores.

En este trabajo se presenta las ecuaciones correctas de la cinemática directa del robot actual. En base a las coordenadas mostradas en Fig. 2.21, se desarrolló la ecuación 2.17, la cual es una matriz que presenta el cálculo de los componentes x, y y z de la punta de las patas.

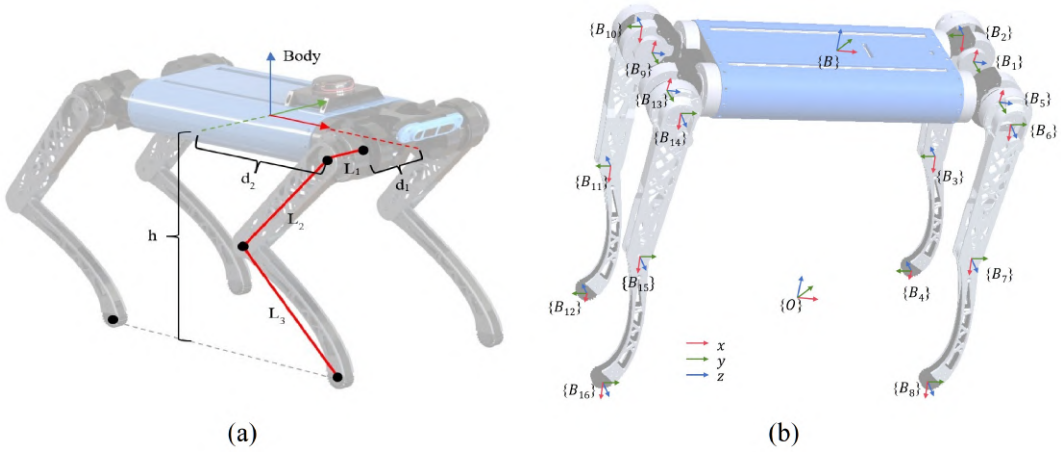


Figura 2.21: Descripción Espacial del Robot. (a) Denominación de los vínculos (b) Numeración de las tramas Recuperado de: [40]

$$^{B}P_{B_i} = \begin{bmatrix} \lambda_i d_2 - L_2 s_2 - L_3 s_{23} \\ \delta_i (d_1 + L_1 c_1) + L_3 s_1 c_{23} + L_2 c_2 s_1 \\ \delta_i L_1 s_1 - L_3 c_1 c_{23} - L_2 C_2 c_1 \end{bmatrix} \tag{2.17}$$

También se presentan dos términos, λ_i y δ_i, los cuales permiten adaptar la matriz a las 4 piernas. Estos dos parámetros dependen del número de trama, el cual se puede consultar en Fig. 2.21.b. La definición de los mismos se puede ver en las ecuaciones 2.18 y 2.19.

$$\lambda = \begin{cases} 1 & \text{cuando } i = 4, 12 \\ -1 & \text{cuando } i = 8, 16 \end{cases} \tag{2.18}$$

$$\delta = \begin{cases} 1 & \text{cuando } i = 4, 8 \\ -1 & \text{cuando } i = 12, 16 \end{cases} \tag{2.19}$$

Por otro lado, en dicho trabajo también se abarcó el desarrollo del jacobiano, el cual se presenta en la ecuación 2.20.

$$J(\theta) = \begin{bmatrix} 0 & -L_2 c_2 - L_3 c_{23} & -L_3 c_{23} \\ \lambda L_1 s_1 + L_2 c_1 c_2 + L_3 c_1 c_{23} & -L_2 s_1 s_2 - L_3 s_1 s_{23} & -L_3 s_1 s_{23} \\ \lambda L_1 c_1 + L_2 s_1 c_2 + L_3 s_1 c_{23} & L_2 c_1 c_2 + L_3 c_1 s_{23} & L_3 c_1 s_{23} \end{bmatrix} \tag{2.20}$$

Capítulo 3

Metodología

La metodología empleada en este trabajo se trata de la expuesta por Karl T. Ulrich y Steven D. Eppinger en su libro "Diseño y desarrollo de productos" [37]. Se seleccionó esta debido a la facilidad de aplicarla a situaciones en donde se le deba brindar un servicio ingenieril a un cliente con ciertas necesidades. La misma está segmentada por una serie de etapas, las cuales se presentan en Fig. 3.1

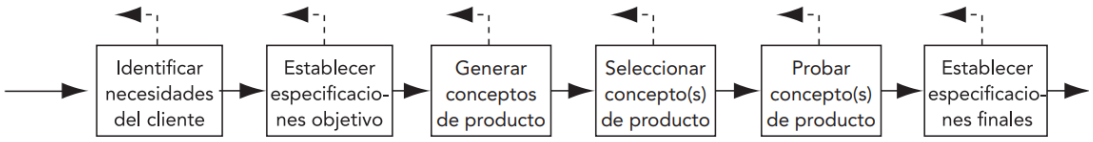


Figura 3.1: Etapas de la metodología. Adaptado de: [37, p. 16].

3.1 Identificación de las necesidades del cliente

Esta etapa tiene como objetivo agilizar la comunicación entre el cliente y el ingeniero para lograr una clara definición de los requisitos del producto [37, p. 74]. Para ello, inicialmente se recopilan datos del cliente a través de una entrevista con el director del laboratorio. En ella, se desarrollaron una serie de preguntas que buscaban abarcar los diversos aspectos del problema propuesto. De esta manera se obtuvo los enunciados del cliente, respaldados mediante una grabación de la misma.

Posteriormente se debe realizar una interpretación de estos datos, en donde se logre encontrar las necesidades relacionadas a los enunciados del cliente. Para ello se siguen las siguientes recomendaciones, recopiladas de [37, p. 83]:

- La necesidad se debe enfocar en la función del producto y no en cómo se va a lograr la misma.
- Se mantiene el nivel de detalle de la información del enunciado del cliente.

- Se utiliza enunciados en forma afirmativa.
- Se utilizan atributos del producto para expresar la necesidad.
- Se evita la palabra "deber" al describir la necesidad, ya que no aún no se conoce su importancia.

Luego se agrupan las necesidades interpretadas de acuerdo con su semejanza. Las necesidades que hablan acerca de algún detalle en específico podrán considerarse secundarias o terciarias, mientras que las que abarquen un aspecto en general se considerarán primarias. De esta manera, se espera encontrar un conjunto de necesidades primarias, las cuales engloben más necesidades secundarias y terciarias.

Después de ello, se establece una importancia relativa a cada necesidad. Para ello se emplea la escala presentada en la tabla 3.1. Esto debe ser evaluado por el cliente y constituye de otra herramienta para clarificar su deseo con el producto. Además, resulta de gran ayuda para priorizar o descartar conceptos, en base a la manera en que satisfacen las necesidades más importantes. Este proceso fue realizado mediante un formulario en línea, con el fin de agilizar la comunicación y simplificar el proceso debido a la gran cantidad de necesidades.

Calificación	Evaluación
1	La función no se desea en el producto.
2	La función no es muy relevante.
3	La función no es necesaria, pero sería conveniente que el producto la tuviese
4	La función es de gran importancia.
5	La función es indispensable.

Tabla 3.1: Escala de necesidades

3.2 Establecimiento de las especificaciones objetivo

Las especificaciones consisten en describir con más detalle lo que el producto tiene que hacer. Estas se componen de una métrica y valores objetivo. Primero se debe determinar la lista de las métricas, las cuales se derivan directamente de las necesidades, en donde en algunos casos se requiere de más de una métrica para abarcar la extensión de una necesidad. Posteriormente se debe seleccionar los valores objetivos, los cuales constituyen de uno ideal y uno marginal. El valor marginal corresponde al mínimo con lo que debe cumplir el producto, mientras que el ideal es una meta realista de lo mejor que se puede esperar del mismo [37, p. 103]. Dichos valores se determinan de acuerdo con los enunciados del cliente expuestos en la entrevista, al comparar productos similares o al realizar una búsqueda externa.

3.3 Generación de conceptos de producto

Construir los conceptos del producto consiste en sintetizar hasta cierto punto, propuestas del funcionamiento, apariencia física o tecnología del producto [37, p. 120]. Primeramente, se realiza una descomposición funcional del problema central, lo cual requiere de interpretar el problema como una caja negra que posee entradas y salidas de material, energía y, señales, tal como se ve en Fig. 3.2. En base a ello, se divide la caja negra y se determinan distintas subfunciones. Este proceso se realiza hasta que las subfunciones sean lo suficientemente sencillas como para poder resolverlas directamente [37, p. 123].

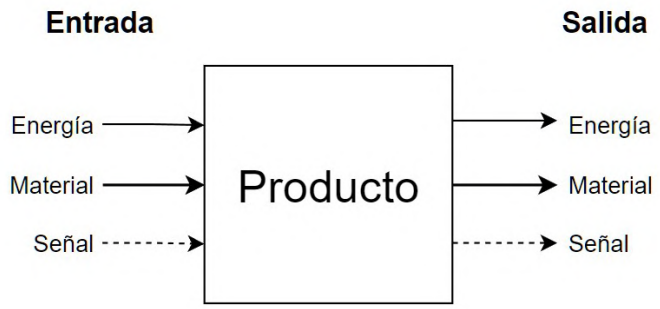


Figura 3.2: Descomposición funcional del problema. Adaptado de: [37, p. 124].

Una vez se tienen los distintos subproblemas a los cuales se les deben proponer soluciones, se realiza una búsqueda externa, que abarca patentes, literatura, comparación de productos relacionados, entre otros. También se realiza una búsqueda interna, con el propósito de hacer uso de los conocimientos previos del ingeniero, así como de su creatividad. Los conceptos generados en este paso son estructurados en árboles clasificadores, los cuales permiten agrupar el conjunto de soluciones de acuerdo con distintas categorías.

Por último, se toman los conceptos generados para cada subproblema y se escriben en una tabla, para luego realizar distintas combinaciones entre ellos. De esta manera, se van a obtener distintas propuestas de solución que solventan cada uno de los aspectos del problema inicial.

3.4 Selección de conceptos de producto

El resultado de la etapa anterior es una serie de propuestas de solución, las cuales serán evaluadas en esta parte de la metodología. Para ello, primeramente se someten a un filtrado de conceptos, en el cual se determina rápidamente si algunas de las propuestas merecen ser combinadas o descartadas. Para ello se emplea una matriz de filtrado, en la cual se emplea una serie de criterios de selección, correspondientes a las métricas más relevantes. La evaluación en este proceso consiste en asignar "+" (mejor), "-" (peor) o "0" (igual) a los conceptos en cada uno de los criterios. Después, se obtendrá una calificación

neta, a partir de la cual se decide si descartar, combinar o continuar con el concepto, en base al resultado de los demás [37, p. 149].

Posterior al filtrado, se realiza una evaluación de conceptos, la cual brinda una comparativa más detallada de las soluciones. Para ello se emplea la escala presentada en la tabla 3.2. Por otro lado, también se emplea el peso de los criterios, ya que, como se demostró en las primeras etapas de la metodología, las necesidades poseen importancias distintas, por lo cual se deben priorizar las sugeridas por el cliente. En base a estos criterios de evaluación, se tabula una matriz de selección con las propuestas de solución. A partir de los resultados se podrá determinar si es posible combinar o mejorar alguno de los conceptos.

Desempeño relativo	Calificación
Mucho peor que la referencia	1
Peor que la referencia	2
Igual que la referencia	3
Mejor que la referencia	4
Mucho mejor que la referencia	5

Tabla 3.2: Escala para la evaluación de conceptos. Recuperado de: [37, p. 155]

Después de haber realizado las iteraciones necesarias, se podrá obtener el concepto ganador, el cual tendrá el puntaje más alto en la matriz de selección. El mismo será puesto a prueba en la siguiente etapa.

3.5 Probar conceptos de producto

Las propuestas de solución seleccionadas en etapa anterior deberán ser sometidas a una serie de pruebas, en las cuales se validarán distintos aspectos de su diseño. El desarrollo de estas pruebas se encuentra con mayor detalle en el capítulo 4, en donde se toman en cuenta detalles propios de dicha solución para desarrollar dicha validación.

3.6 Establecer especificaciones finales

Una vez se tenga un concepto seleccionado, caracterizarlo y mostrar sus especificaciones finales. A partir de ello, se observará si la solución desarrollada cumple con los requisitos planteados desde un principio y si es necesario realizar iteraciones adicionales. En el capítulo 5 se desarrollan las pruebas de la implementación del conjunto de soluciones y además, se determina el rendimiento del producto final.

Capítulo 4

Desarrollo de la solución

En este apartado se presenta el proceso de toma de decisiones para resolver las necesidades del cliente. Primeramente, se realiza un diagnóstico del problema, con el fin de proporcionar un mejor contexto de la arquitectura previa y posteriormente se muestra el desarrollo de los pasos mostrados en la metodología.

4.1 Principales diferencias entre la realidad y la simulación

En el primer enunciado del problema expuesto por el cliente, se mencionó que al tratar de ejecutar un algoritmo de control en una de las piernas, el lazo de control presentaba una frecuencia muy baja, de alrededor de 50 Hz. Además, se comentó que el problema correspondía a la comunicación con los actuadores, por lo que requerían de una mejora de hardware.

El diagrama del experimento desarrollado es mostrado en Fig. 4.1, en cual se realiza un control de posición de una de las piernas, empleando un lazo de realimentación en la computadora central. De esta manera, se envían los comandos de torques a los 3 actuadores de la pierna y además, se lee su posición.

Para corroborar los datos presentados en el enunciado, se realiza una réplica del experimento, los resultados se presentan en la tabla 4.1. En esta se observa que inicialmente se tiene una frecuencia de control muy similar a la establecida por el cliente. Sin embargo, se detectó que el código tenía múltiples impresiones de variables para depurar el proceso, lo cual resulta en retrasos significativos. En base a ello, se omiten dichas impresiones y se repite el experimento, en donde se observa que la frecuencia aumentó hasta 164.66 Hz. No obstante, este corresponde solo al control de una de las patas, ya que al ejecutar el control en las 4 piernas, se obtuvo una frecuencia de tan solo 40.16 Hz.

Dicho desempeño en la frecuencia de control representa un gran impedimento para implementar los algoritmos de control desarrollados anteriormente. Como referencia, el control

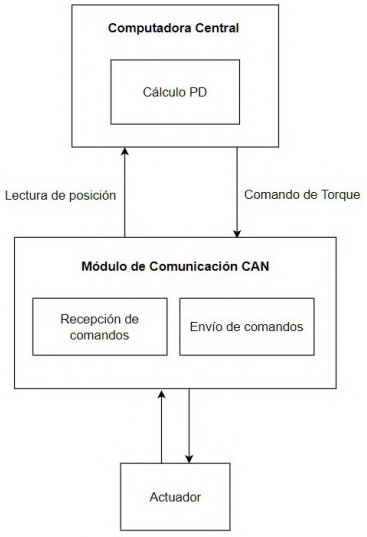


Figura 4.1: Diagrama del funcionamiento del control PD. Elaboración Propia

dinámico MPC, previamente elaborado, emplea una frecuencia de 200 Hz, e incluso existen otros algoritmos que requieren tasas superiores a 500 Hz.

Experimento	Frecuencia promedio (Hz)
Código por defecto	50.64
Código sin impresión de variables	164.66
Control en las 4 piernas	40.16

Tabla 4.1: Resultados de frecuencia de control del sistema inicial

Otra de las mayores diferencias encontradas con respecto a las simulaciones desarrolladas, corresponden a la falta de una detección del contacto y una estimación de estados del centro de masa. Estos elementos son clave en el funcionamiento de las secuencias de caminata y además, corresponden a módulos importantes en la mayoría de las arquitecturas de control de robot cuadrúpedos.

La estimación de posición empleada en las simulaciones corresponde a un sensor GPS, sin embargo, las capacidades de este componente en la vida real resultan insuficientes para ser incorporadas en un algoritmo de control. Si bien corresponde a un problema principal, actualmente uno de los miembros del laboratorio se encuentra desarrollando un estimador de estados empleando la odometría y un filtro de Kalman. Por esto, no se considerará en el desarrollo de la solución, pero se propondrá una versión relativamente simple para evaluar el funcionamiento de la arquitectura.

Por otro lado, el modelo empleado resulta en una gran simplificación con respecto al modelo real, esto en términos mecánicos. En Fig. 4.2 se aprecia dicho modelo, en donde, por ejemplo, se puede observar como los vínculos de las piernas consisten en cilindros. Por lo cual, posiblemente exista una diferencia en el momento de inercia con respecto a la realidad, lo cual se traducirá en un comportamiento dinámico distinto. Asimismo, los

actuadores empleados en la simulación poseen su propio control PID interno, es decir, a pesar de que los comandos enviados por la computadora central en la realidad y la simulación sean los mismos, el comportamiento de los actuadores será distinto. Dichas diferencias se verán reflejadas en las pruebas funcionales, sin embargo, no se pretende aumentar la fidelidad del modelo, debido a que este es un tema en el que el laboratorio planea trabajar posteriormente a la mejora de la arquitectura.

Figura 4.2: Simplificación del modelo en la simulación. Elaboración Propia

Por otro lado, es importante mencionar la propia interfaz de simulación, la cual posee funciones absolutas para detener, pausar, reiniciar o correr el algoritmo. Dichos comandos son ejecutados e interpretados de manera inmediata, lo cual actualmente no se posee en el robot real.

4.2 Necesidades y Especificaciones

Las necesidades del proyecto fueron determinadas a partir de una entrevista con el cliente, en donde se realizó una serie de preguntas relevantes al problema y en base a ello se sintetizó la información más importante para establecer las necesidades, las cuales se pueden observar en la tabla 4.2. En una entrevista posterior, se le presentaron las necesidades encontradas al cliente y este les asignó importancia, que también se incluye en la tabla.

Después de ello, se desarrollaron las respectivas métricas para determinar el nivel de cumplimiento de las necesidades. El nivel de importancia de cada métrica viene dado por el nivel de importancia de la necesidad que se busca evaluar. Por otro lado, también se definieron los valores objetivo, esto mediante los resultados encontrados en la entrevista, así como una búsqueda externa, esto se puede observar en el apéndice A.1.2. Las métricas que hace referencia a una escala subjetiva consisten en pruebas de satisfacción que serán realizadas por el usuario final, de manera que se puede llegar a elegir un 5 para indicar el máximo cumplimiento de la solución o la asignación de un 1 para indicar que esta resultó deficiente.

Núm.	Necesidades Interpretadas	Imp.
1	**El sistema diseñado (SD) se comunica adecuadamente con el hardware.**	
1.1	El sistema diseñado lee los sensores correctamente.	3
1.2	El SD le proporciona la información necesaria al controlador.	4
2	**EL SD interactúa apropiadamente con el robot.**	
2.1	El sistema es fácilmente controlable por el usuario.	5
2.2	El robot realiza tareas complejas con instrucciones simples.	4
3	**El SD puede desempeñar una secuencia de caminata pertinente.**	
3.1	Es SD puede ejecutar diferentes secuencias de caminado.	5
3.2	El robot puede moverse a través de varios tipos de terrenos.	4
3.3	El robot se desplaza a una velocidad considerable.	4
3.4	El robot físico se asemeja estrechamente a la simulación.	4
3.5	El robot realiza movimientos fluidos.	4
3.6	El robot puede caminar una distancia considerable.	5
3.7	El cuerpo del robot permanece estable durante los movimientos.	5
4	**El SD posee una arquitectura flexible y un desempeño adecuado.**	
4.1	El SD puede manejar algoritmos de control complejos.	4
4.2	Las funciones del robot respetan una jerarquía de importancia.	4
4.3	El SD es capaz de procesar todos los datos.	3
4.4	El sistema permite una fácil implementación de algoritmos de control.	4
4.5	El SD posee un diseño modular.	4
4.6	La arquitectura de control del SD se puede modificar fácilmente.	5
4.7	El bucle de control del robot se mantiene estable.	5
5	**El SD se mueve de manera segura.**	
5.1	El SD asegura la integridad de sus componentes.	4
5.2	El robot realiza movimientos seguros para los humanos.	4
6	**El robot opera continuamente durante un periodo significativo.**	5
7	**El robot requiere calibración en intervalos óptimos.**	3
8	**El SD es capaz de desarrollarse.**	
8.1	El SD tiene un costo adecuado.	5
8.2	El SD se desarrolla dentro de un marco de tiempo óptimo.	4

Tabla 4.2: Necesidades encontradas del proyecto

4.3 Descomposición en subproblemas

El problema central se dividió en 4 categorías iniciales, correspondientes a una interfaz de pruebas que actúe de manera radical sobre los actuadores, una detección de contacto, una interfaz de comunicación y la manera en que se da el procesamiento de la locomoción. En Fig. 4.3 se observa dicha descomposición, en donde el controlador del motor y la adaptación de la energía están representados con bloques grises debido a que son componentes del problema, pero no se pretenden cambiar en este proyecto.

El bloque de procesar información se decidió subdividirlo en percepción, planeamiento y ejecución, los cuales corresponden a las 3 tareas principales de un robot móvil [16]. Mediante el mismo, se seleccionará la manera más apropiada de validar la arquitectura y ejecutar una de las secuencias de caminata en la etapa actual de desarrollo. En Fig. 4.4 se puede apreciar dicha subdivisión, en donde la parte de percepción visual no será

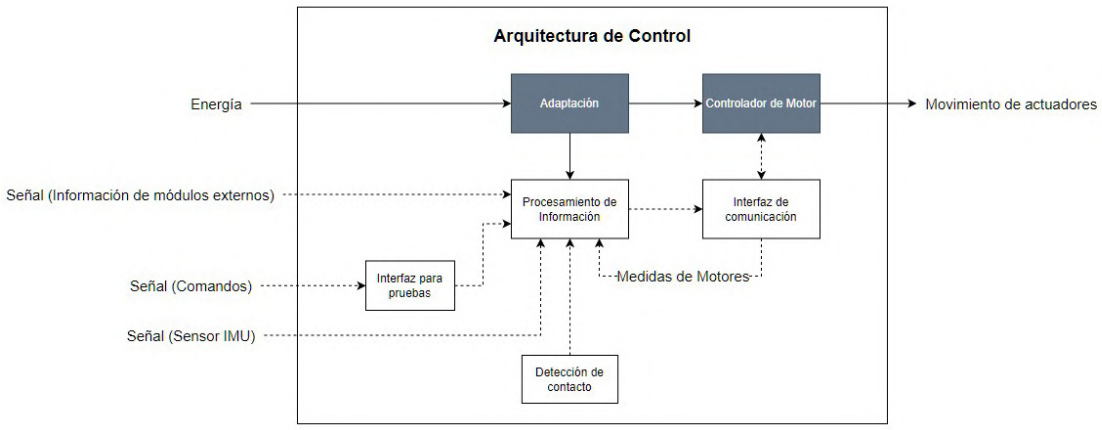


Figura 4.3: Descomposición general del problema. Elaboración propia: Diagrams.net

estudiada debido a que se sale del alcance de este proyecto. Sin embargo, el planeamiento y la ejecución son de vital importancia ya que corresponden a la secuencia de caminata y a la locomoción. Si bien también se posee una percepción mediante los sensores de los actuadores y el sensor de inercia, esto se considera en la interfaz de comunicación.

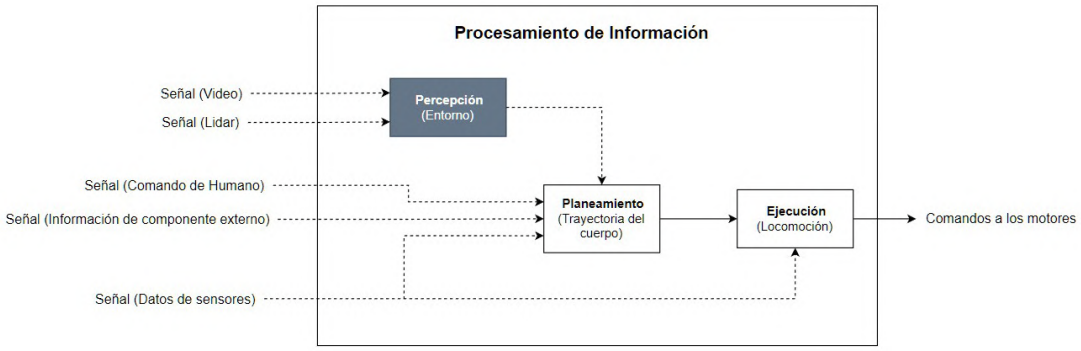


Figura 4.4: Descomposición del subproblema de procesamiento. Elaboración propia: Diagrams.net

Debido a la complejidad de la secuencia de caminata, se decidió subdividir el problema de planeamiento en el tipo de secuencia de caminata y la trayectoria que realiza las patas al balancearse, tal como se ven en Fig. 4.5. Por último, también se subdividió el problema de la interfaz de comunicación, debido a que la componente de hardware de este proceso puede resultar en una complejidad considerable. En Fig. 4.6 se puede apreciar cómo se interpreta la interfaz como una configuración de hardware y de una topología para el gestionamiento de la comunicación.

4.4 Diseño de la interfaz de comunicación

En esta sección se presenta el desarrollo de la metodología para diseñar la interfaz de comunicación, además, se evalúan métodos para mejorar el prototipo seleccionado y alcanzar

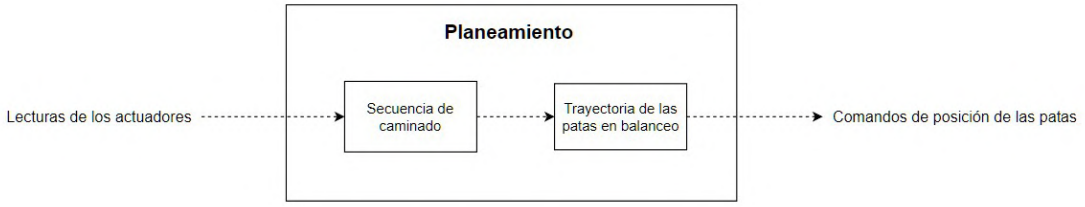


Figura 4.5: Descomposición del subproblema de planeamiento. Elaboración propia: Diagrams.net

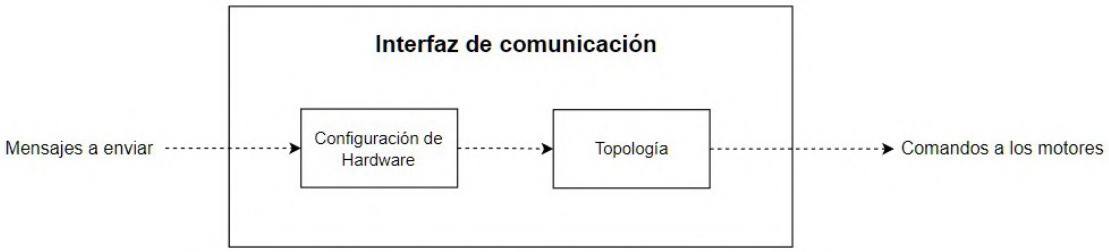


Figura 4.6: Descomposición del subproblema de Interfaz de Comunicación. Elaboración propia: Diagrams.net

el mejor desempeño.

4.4.1 Generación de conceptos de la interfaz de comunicación

En la configuración de hardware se presentan dos clases de arreglos, el de módulo único, en el cual ya tiene controlador CAN y un transceptor integrados y también existe la "combinación", en donde estos dos se encuentran por separado. En Fig. 4.7 se observa las propuestas de solución para este problema, las cuales poseen un diferente protocolo de comunicación e incluso el uso de microcontroladores.

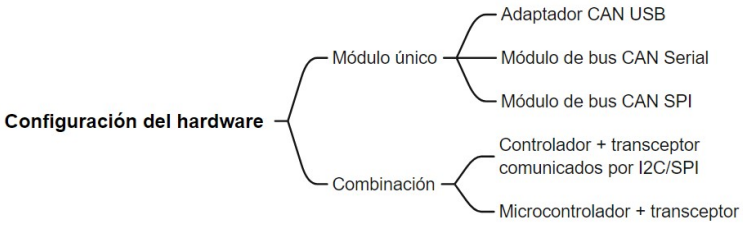


Figura 4.7: Generación de conceptos para la configuración de hardware de la interfaz de comunicación. Elaboración propia: Xmind

La topología de comunicación habla de cómo se estructuran las conexiones de los 12 actuadores y por ende, el gestionamiento de los mensajes. En Fig. 4.8 se presentan los conceptos en este apartado, en donde existe una solución simple de conectar el módulo de comunicación directamente al computador o en caso contrario, usar microcontroladores como puntos medios para coordinar los mensajes.

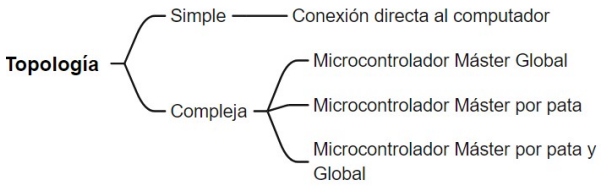


Figura 4.8: Generación de conceptos para la topología de la interfaz de comunicación. Elaboración propia: Xmind

4.4.2 Conceptos de la Interfaz de Comunicación

A continuación, se presentarán las combinaciones de las soluciones de cada problema, para así generar los distintos conceptos, esto en base a las soluciones propuestas en la etapa anterior. En la totalidad de ellos se asumen 4 redes de comunicación, esto se obtiene al analizar el hardware de otros robots cuadrúpedos y además, posteriormente se muestra el desempeño máximo de 2 redes CAN en el robot, el cual resultó ser insuficiente para los requisitos del cliente.

Concepto A

El primer concepto consiste en el empleo de un adaptador CAN-USB como el hardware y además, presenta una topología simple, ya que se conecta directamente al computador, esto se observa en Fig. 4.9. El boceto del concepto A se puede apreciar en Fig. 4.10, en

Figura 4.9: Construcción del Concepto A. Elaboración propia: Diagramas.net

la cual se observa como el módulo CAN se conecta a la computadora central mediante un puerto USB y posteriormente, este módulo se comunica con 3 de los actuadores del robot, es decir, con los de una de las patas, lo cual implica el uso de 4 módulos.

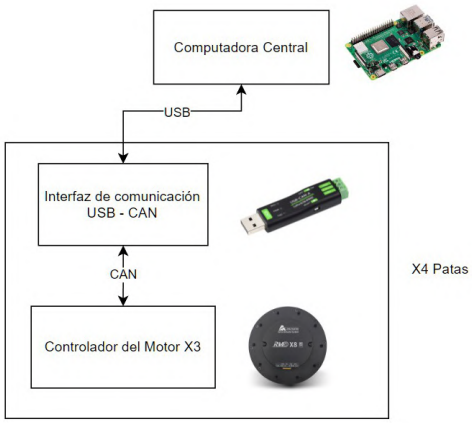


Figura 4.10: Boceto del Concepto A. Elaboración propia: Diagramas.net

Concepto B

Por otro lado, el concepto B emplea un Módulo CAN que se comunica por medio del protocolo serial y además, emplea un microcontrolador para gestionar los mensajes de cada una de las patas y otro para gestionar la totalidad de los mensajes con la computadora central, tal como se observa en Fig. 4.11.

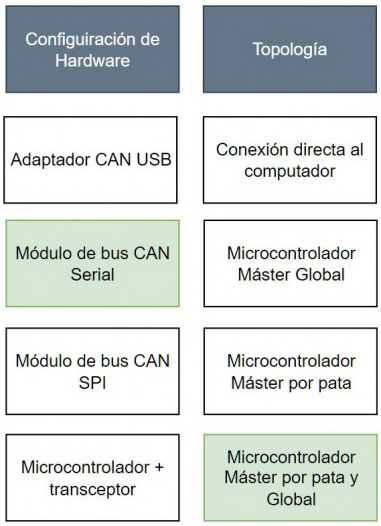


Figura 4.11: Construcción del Concepto B. Elaboración propia: Diagramas.net

La topología de este concepto se puede observar en Fig. 4.12, en donde es importante mencionar que al emplear microcontroladores intermedios no solo se tiene el beneficio de gestionar los mensajes rápidamente, sino que también se pueden implementar niveles bajos de los algoritmos de control a frecuencias altas, como lo puede ser un control de fuerza resultante para cada una de las patas.

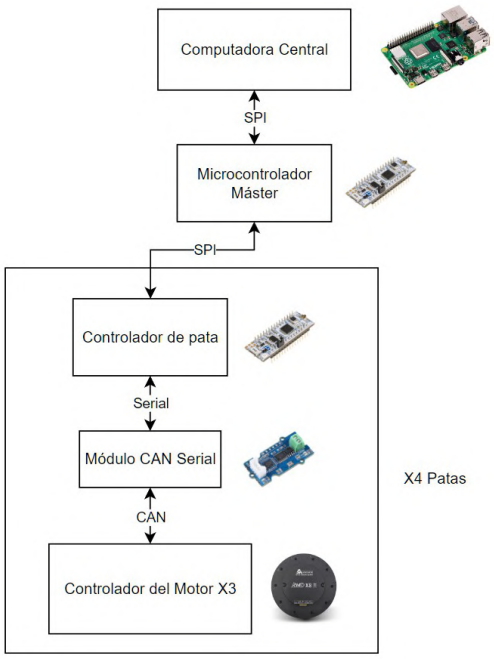


Figura 4.12: Boceto del Concepto B. Elaboración propia: Diagramas.net

Concepto C

El concepto C consiste en emplear un microcontrolador, el cual posee un periférico de comunicación CAN, en conjunto con un transceptor. Finalmente, el microcontrolador está conectado directamente a la computadora central, esto se muestra en Fig. 4.13. Por otro lado, en Fig. 4.14 se presenta la topología de la red de comunicación. Algunos microcontroladores poseen más de un periférico de comunicación CAN, por lo que el número de microcontroladores requeridos en este concepto no es 4. En este caso, la señal de comunicación serial empleada transceptor es propia del controlador CAN que posee el microcontrolador.

Concepto D

El concepto D se compone de módulos CAN comunicados por medio de SPI y además, emplea un microcontrolador máster para cada una de las patas, de la manera en que se ve en Fig. 4.15. El boceto de este concepto se presenta en Fig. 4.16 en donde se aprecia un módulo CAN que maneja un canal y se comunica por medio de SPI con el controlador de la pata individual, posteriormente los cuatro controladores se comunican con la computadora central por medio de SPI.

Figura 4.13: Construcción del Concepto C. Elaboración propia: Diagramas.net

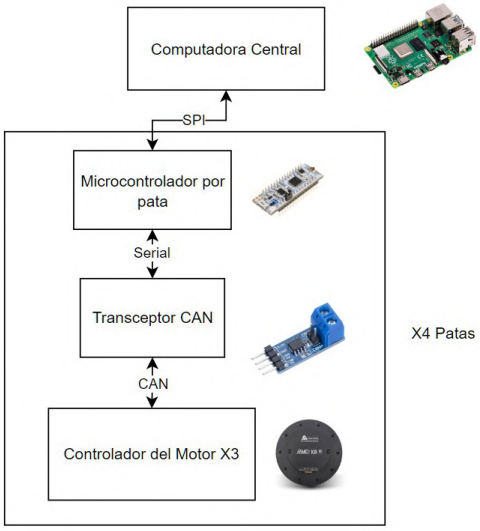


Figura 4.14: Boceto del Concepto C. Elaboración propia: Diagramas.net

4.4.3 Selección de los conceptos de la Interfaz de Comunicación

En la tabla 4.3 se desarrolló el filtrado de los conceptos de la interfaz de comunicación. En esta se puede apreciar como en concepto B obtuvo una calificación muy baja en comparación con los demás, por lo cual se tomó la decisión de cambiar alguno de sus aspectos. Al observar los criterios en donde no destaca de los demás, se puede inferir que el cambio que se va a realizar deber tener mejoras en la modularidad, facilidad de modificación o el costo económico.

A partir de ello, se propone realizar un cambio en su topología de comunicación. Inicialmente se planteó tener un microcontrolador como un máster global que se comunicaba con la computadora central, además de tener 4 microcontroladores que controlaran los mensajes de cada una de las piernas del robot. Este corresponde al principal motivo por el

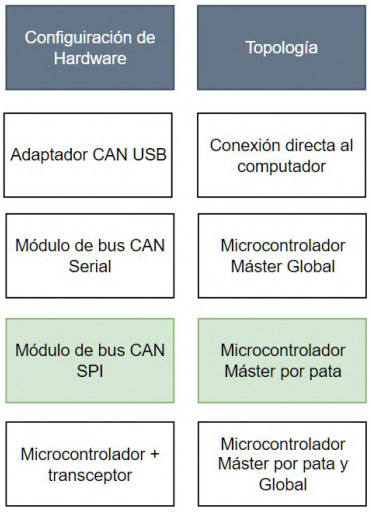


Figura 4.15: Construcción del concepto D. Elaboración propia: Diagramas.net

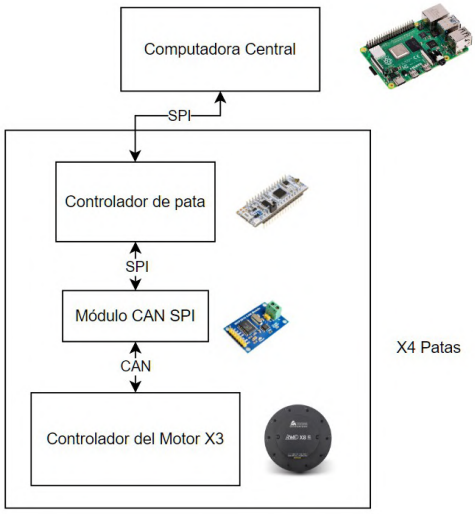


Figura 4.16: Boceto del Concepto D. Elaboración propia: Diagramas.net

cual el candidato se vio desfavorecido en la matriz de filtración. Por lo tanto, se propone una topología de un único microcontrolador, el cual será el máster Global, eliminando 4 de los microcontroladores que ocasionaban un aumento en el costo, el consumo de energía y la dificultad para realizar cambios en la arquitectura. Con base en esto, se desarrolla en boceto de concepto B.2, el cual se puede apreciar en Fig. 4.17

Una vez se ha realizado el filtrado de los conceptos, se realiza la selección final, la cual se presenta en la tabla 4.4. En esta selección se puede ver que los conceptos “A” y “B.2” obtuvieron las calificaciones más altas. Además, se observa que el concepto “A” fue superior en modularidad, facilidad de modificación y tiempo de desarrollo, los cuales poseen un gran peso en la matriz de decisiones. Si bien el concepto B.2 puede resultar ligeramente mejor en rendimiento del sistema de comunicación, sus desventajas en los demás aspectos importantes muestran que dicha mejora no vale la pena de acuerdo con los requerimientos de este proyecto. Debido a esto, se pretende desarrollar la solución

Criterios de selección	Conceptos			
	A	B	C	D
Rendimiento	+	+	+	+
Modularidad	0	-	0	0
Facilidad de modificación	+	-	+	+
Consumo de energía	0	-	0	0
Costo	0	-	0	0
Tiempo de desarrollo	+	0	0	0
Suma +	3	1	2	2
Suma 0	3	1	4	4
Suma -	0	4	0	0
Evaluación Neta	3	-3	2	2
Lugar	1	4	2	2
¿Continuar?	Sí	Cambiar	Sí	Sí

Tabla 4.3: Matriz de Filtrado para los conceptos de la Interfaz de comunicación

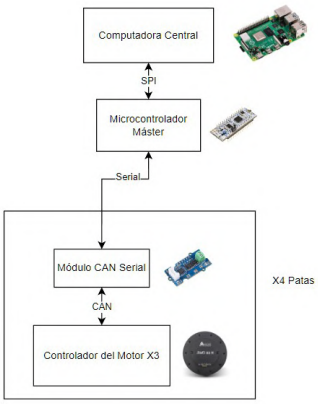


Figura 4.17: Boceto del Concepto B.2. Elaboración propia: Diagramas.net

descrita por el concepto “A”.

Criterios de selección	Peso (%)	Conceptos							
		A		B.2		C		D	
		C.	E.	C.	E	C.	E	C.	E
Rendimiento	15.4	3	0.46	4	0.62	3	0.46	4	0.62
Modularidad	15.4	5	0.77	4	0.62	3	0.46	4	0.62
Facilidad de modificación	19.2	5	0.96	4	0.77	3	0.58	3	0.58
Consumo de energía	15.4	4	0.62	4	0.62	3	0.46	3	0.46
Costo	19.2	3	0.58	3	0.58	3	0.58	2	0.38
Tiempo de desarrollo	15.4	5	0.77	4	0.62	3	0.46	3	0.46
Total de puntos	100.0	24	4.2	23	3.8	18	3.0	19	3.1
Lugar		1		2		4		3	
¿Continuar?		Sí		No		No		No	

Tabla 4.4: Matriz de selección para la Interfaz de comunicación

4.4.4 Implementación de la Interfaz de comunicación

Primeramente, se analizarán posibles mejoras en el aspecto del software y la lógica de la comunicación. En Fig. 4.18 se puede observar el proceso por el cual se comunicaba la computadora en un inicio. En esta se puede observar que es necesario enviar un comando de torque al actuador y después de un tiempo este responderá automáticamente con los datos actuales de posición, velocidad y torque.

Cabe recalcar que esta posición es la del ángulo del eje del motor, sin embargo, este posee una relación de engranes con la salida de 9 vueltas. Por otro lado, dicho valor de posición va desde 0 hasta 360°y cuando se sobrepasa el valor máximo, vuelve a 0°. Por lo tanto, no se puede concluir la posición actual de la articulación del robot con esta medida. Es por ello que también se le solicitaba al motor la lectura del ángulo multi-giro, la cual es calculada en el controlador interno del motor y solventa los problemas mencionados.

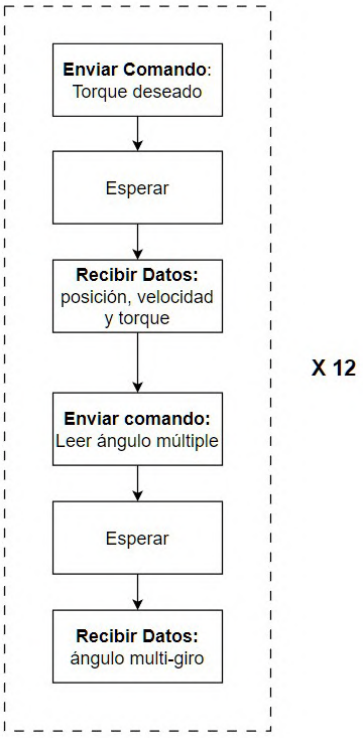


Figura 4.18: Esquema Inicial de la comunicación. Elaboración propia: Diagrams.net

Si bien esta solución resulta efectiva desde un punto de vista funcional, al evaluar el rendimiento del sistema de comunicación, se puede inferir que al requerir enviar un mensaje extra, se duplicarán los tiempos de espera. Es por esto que conviene evaluar la posibilidad de calcular el ángulo multi-giro en la computadora central.

4.4.5 Cálculo del ángulo multi-giro

Para entender este cálculo, se debe asumir que los valores del encoder estarán presentes en el rango de [0 a 360°] en el eje del motor, o el equivalente en el eje de salida que es de [0 a 40°], de la manera en que se presenta en Fig. 4.19. En ella se observa que el ángulo multi-giro está compuesto por la suma de múltiples vueltas del motor, es decir, los bloques de 40°.

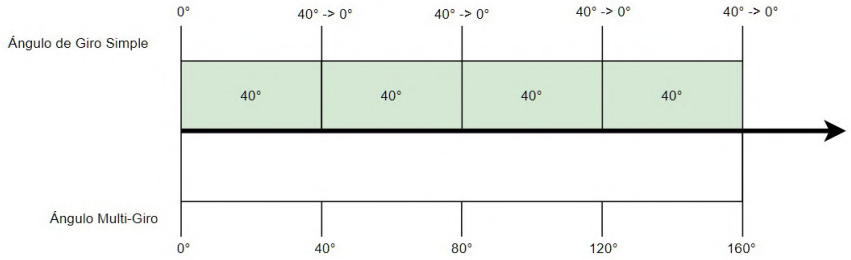


Figura 4.19: Diagrama del multi-ángulo. Elaboración propia: Diagrams.net

A partir de ello, se sabe que la posición absoluta o el multi-ángulo, estará dada por la posición actual del ángulo simple, más las vueltas que este ha realizado anteriormente, tal como se expresa en la ecuación 4.1.

$$\text{Angulo}_{multi-giro} = \text{Angulo}_{simple} + n_{vueltas} \cdot 40^\circ \tag{4.1}$$

Asimismo, se sabrá la dirección de giro al observar los cambios bruscos de la posición. De esta manera, se formula la ecuación 4.2, la cual establece cuándo se debe sumar o restar una vuelta a la posición absoluta del motor.

$$n = \begin{cases} +1 & \text{cuando } \theta = 40^\circ \rightarrow \theta = 0^\circ \\ -1 & \text{cuando } \theta = 0^\circ \rightarrow \theta = 40^\circ \end{cases} \tag{4.2}$$

Por otro lado, es importante determinar si este cálculo resulta confiable, considerando la velocidad del motor y la tasa de lectura del actuador. La velocidad del motor sin carga a 24V corresponde a 960rpm, lo cual es equivalente a 16 rev/s. Es decir, el motor es capaz de girar a 16 Hz como máximo, ya que en la práctica tendrá una carga. En base al teorema de Nyquist, la frecuencia de muestreo requerida para digitalizar una señal analógica debe ser de al menos el doble de esta [4]. En este caso, se considera el movimiento del motor a la velocidad máxima como dicha señal, de manera que se plantea la ecuación 4.3.

$$f_{\text{muestreo}} \geq 2 \cdot f_{\text{señal}} = 2 \cdot 16\,\text{Hz} = 32\,\text{Hz} \tag{4.3}$$

De esta manera, se determina que la frecuencia mínima para detectar cambios en las revoluciones del motor será de 32 Hz. Por lo tanto, al tomar en cuenta que la frecuencia

de muestreo de los motores es mucho mayor que esta, se puede concluir que es físicamente posible medir los cambios en las vueltas en los motores para así, calcular el ángulo multigiro desde el computador central. Al eliminar la necesidad de la lectura del ángulo multigiro, la frecuencia de muestreo aumenta a un 200% del valor inicial.

4.4.6 Aprovechamiento de los canales

En Fig. 4.20 se puede ver el esquema inicial del uso de los canales, en donde se procesaban los mensajes de los 6 actuadores del primer canal, para después proceder a los del otro canal.

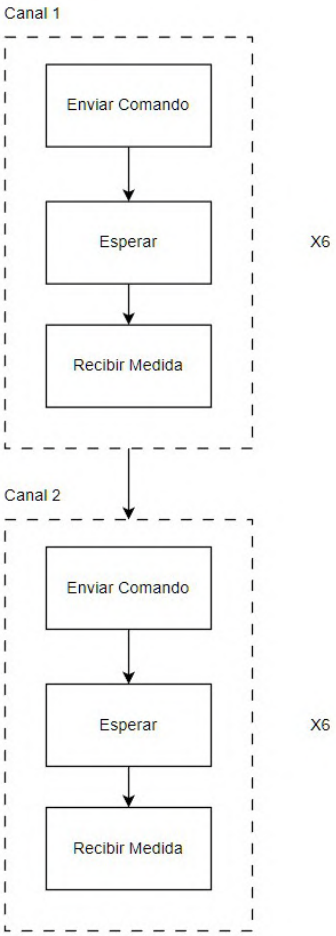


Figura 4.20: Esquema Inicial de la comunicación. Elaboración propia: Diagrams.net

A partir de ello, se propone emplear los canales de forma paralela. En este sentido, se enviarán dos mensajes a la vez, uno en cada canal, y las respuestas de ambos se recibirán seguidamente del tiempo de espera normal, de la manera en que se presenta en el diagrama de Fig. 4.21

Al realizar las pruebas con esta configuración, se observa que la media de la frecuencia de muestreo es de alrededor de 215.4 Hz.

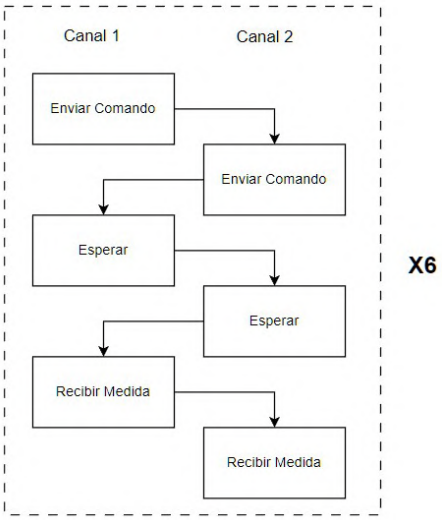


Figura 4.21: Aprovechamiento de los canales. Elaboración propia: Diagrams.net

4.4.7 Rendimiento máximo del canal

Por otro lado, se desea observar el rendimiento máximo de cada canal. Para ello se realiza una prueba en la cual se trata de enviar 6 mensajes de manera seguida y observar cómo la computadora central recibe cada una de sus respuestas. En Fig. 4.22 se puede ver el resultado que se muestra en la consola del computador. En este caso se está empleando la herramienta de visualización de mensajes CAN de Linux. En esta figura se espera ver una serie de mensajes con la identificación (ID) desde 0x141 hasta 0x146, representando a los motores. Los datos de los mensajes poseen 8 bytes, compuestos por 8 parejas de números hexadecimales. El primer byte consiste en el identificador del comando empleado, correspondiente a "9C". Las peticiones de lectura de la computadora central poseen los 7 bytes subsecuentes al comando como "00", mientras que los que poseen algún dato diferente de 00, constituyen las respuestas de los actuadores.

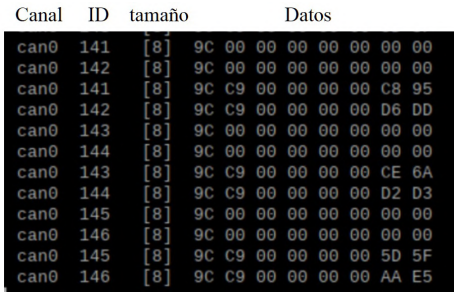


Figura 4.22: Prueba de respuesta de la comunicación. Elaboración propia: Diagrams.net

Es importante respetar las prioridades del protocolo de comunicación. Es decir, los mensajes se deben enviar en orden desde 0x141 hasta 0x146. Esto se debe que se puede ocasionar la pérdida de mensajes en la situación de que dos motores quieran comunicarse al mismo momento, pero el mensaje de prioridad menor estaba primero, por lo que el nuevo mensaje de mayor prioridad lo desplazara. Esto se debe a un conflicto en los búferes de

los mensajes, en donde a partir de pruebas realizadas, se determinó que al tener mensajes salientes y entrantes se puede ocasionar la pérdida de los mensaje de menor prioridad.

Con base en esto, se puede observar que inicialmente se envían dos comandos, hacia los motores 0x141 y 0x142 y posteriormente se recibe ambas respuestas. Después se envían los comandos hacia 0x143 y 0x144 para después obtener su respuesta. A partir de este comportamiento, se puede deducir que es posible enviar dos comandos de manera subsecuente, para después recibir las dos respuestas, todo esto dentro del mismo canal.

Por lo tanto, se propone una lógica en la comunicación, en donde se envían dos mensajes dentro de un mismo canal de manera seguida, y a su vez, también se haga esto en el otro canal, de manera que se envían 4 mensajes en total, para después esperar un ligero tiempo y recibir las 4 respuestas de los actuadores. El diagrama que resume la lógica de este método presenta en Fig. 4.23.

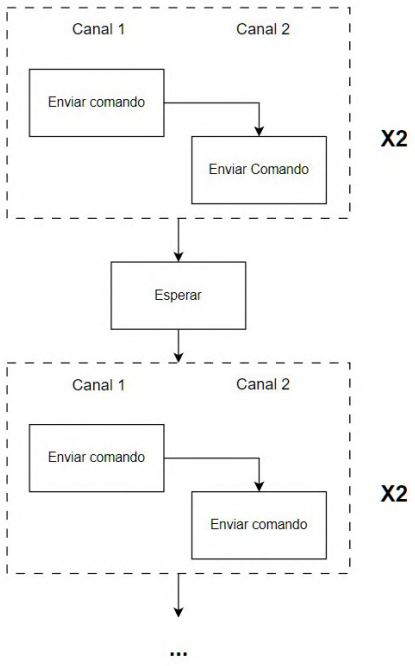


Figura 4.23: Aprovechamiento de los canales. Elaboración propia: Diagrams.net

Al realizar las pruebas de rendimiento de este método, se obtuvo un promedio de frecuencia de muestreo de 282.3 Hz.

Dichas pruebas fueron realizadas en el lenguaje de Python, y al desarrollar las librerías y programación necesarias en el lenguaje C++, se obtuvo una frecuencia de alrededor de 374.8 Hz, demostrando su superioridad. En la tabla 4.5 se presenta el resumen de las mejoras propuestas en este apartado.

Las pruebas para obtener los resultados de la media de frecuencia de muestreo se realizaron al procesar 1000 ciclos, debido a que mostraba ser una medida consistente al repetirse en múltiples ocasiones. En el siguiente capítulo se indagará más en la estadística para evaluar el rendimiento real del sistema.

Solución	Frec. muestreo (Hz)	Porcentaje del valor original (%)
Original	64.8	100.0
Sin ángulo multi-giro	126.8	195.7
Paralelismo de canales	215.4	332.4
Mensajes dobles por ciclo	282.3	435.6
Lenguaje C++	374.8	578.4

Tabla 4.5: Resumen del desempeño de la lógica de comunicación

Si bien en esta sección se presenta un método para mejorar substancialmente el rendimiento del sistema de comunicación empleando únicamente el software de la computadora central, el cliente expresó que le gustaría poder llegar a duplicar este resultado, con el evitar futuros cuellos de botella cuando se requiera investigar nuevos algoritmos de control que demanden una mayor frecuencia de muestreo. Por lo cual, se debe recurrir al hardware y duplicar el número de canales, tal como se plantea en los conceptos propuestos.

Frecuencia de muestreo máxima

Cuando se trabaja en el diseño de redes CAN, en las cuales se comunican múltiples dispositivos periódicamente, se recomienda que el flujo de los mensajes no supere el 30% de la capacidad del bus de transmisión, esto con el fin de que los mensajes de baja prioridad no experimenten retrasos sumamente grandes [11].

En este caso, se trabaja con CAN 2.0A, el cual posee una velocidad de 1 Mbps. Por lo tanto, el ancho de banda recomendado será de 300 kbps, tal como se muestra en la ecuación 4.4.

$$BW_{recomendado} = 30\% \cdot BW_{max} = 0.3 \cdot 1Mbps = 300kbps \tag{4.4}$$

Los mensajes empleados corresponden al tipo estándar sin identificador extendido, lo cual constituye un total de 115 bits. La cantidad de mensajes que se enviarán al mismo tiempo, tal como se plantea en la lógica anterior, corresponden a 2, en la primera etapa cuando se debe comunicar con el motor 1 y 2. En este caso no se toma en cuenta la totalidad de los mensajes, debido a que la lógica del procesamiento de los mismos consiste en esperar hasta que lleguen al destino, por lo cual el máximo de mensajes en el bus siempre será de 2. Por otro lado, las recomendaciones del diseño de la red mencionan añadir un factor para el peor de los casos, F_{peor}, el cual es igual a 1.1 [11], esto se muestra en la ecuación 4.5.

$$BW_{requerido} = F_{peor} \cdot n_{mensajes} \cdot bits_{mensaje} \cdot f_{muestreo} \tag{4.5}$$

A partir de ello, se plantea la frecuencia de muestreo como incógnita, debido a que no se conoce el rendimiento del producto final, de manera que se pueda ver si esta lógica cumple con los valores objetivos propuestos en un inicio. En base a esto, se despeja la

frecuencia de muestreo, la cual será la máxima admitida para no sobrecargar el canal, esto se observa en la ecuación 4.6.

$$f_{muestreo} = \frac{BW_{recomendado}}{F_{peor} \cdot n_{mensajes} \cdot bits_{mensaje}} \tag{4.6}$$

En la ecuación 4.7 se determina que dicho valor corresponde a 1186Hz.

$$f_{muestreo} = \frac{300kbps}{1.1 \cdot 2 \cdot 115} = 1186Hz \tag{4.7}$$

Es decir, a la frecuencia máxima en donde se pueden enviar de forma segura 2 mensajes corresponde a un valor mucho mayor al valor ideal planteado en las especificaciones, el cual supera incluso la capacidad de otros robots cuadrúpedos de alto rendimiento, los cuales emplean una comunicación CAN de 1000 Hz. A partir de ello, se concluye que la lógica planteada anteriormente está respaldada por las recomendaciones de diseño y por lo tanto, la misma podrá operar con cierta confiabilidad, sin riesgo de retrasos significativos que afecten notablemente la frecuencia de muestreo general.

4.4.8 Selección de los componentes

El principal requerimiento de los módulos de comunicación es que sean compatibles con CAN 2.0, el cual es la versión que se emplean en los actuadores, con una velocidad de 1 Mbps. Por otro lado, deben poder conectarse por medio de USB a la computadora central y además, poseer un firmware compatible con la funcionalidad de mensajes CAN de Linux.

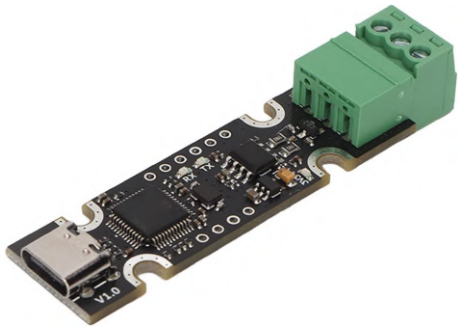


Figura 4.24: Módulo de comunicación CAN-USB. Tomado de: Fysetc.

En base a estos requerimientos, se seleccionó la placa de Estink mostrada en Fig. 4.24, la cual posee el protocolo CAN 2.0B y 2.0A, con una velocidad de transferencia de 1 Mbps. Por otro lado, posee un firmware llamado CANable, el cual es de código abierto y permite emplear las librerías de Linux.

4.4.9 Construcción Física

Para interconectar los 4 módulos de comunicación con la computadora física, se empleó una estación de USB, con el fin de aumentar la modularidad del diseño. Esto se basa

principalmente en las altas velocidades del protocolo USB, el cual puede llegar hasta 480 Mbps en la versión 2.0. Al ser dicha velocidad muy superior a la requerida por los módulos, se considera apropiado conectar los cuatro módulos a solo uno de estos puertos. A partir de ello, se diseña un encapsulado que contenga los 4 módulos, además de 4 adaptadores USB C - USB macho-macho y la estación USB, tal como se observa en el renderizado en Fig. 4.25. En Fig. 4.26 se puede apreciar la construcción física del diseño planteado, en la cual se construye la carcasa mediante impresión 3D.

Figura 4.25: Renderizado de la nueva interfaz de comunicación CAN.

Figura 4.26: Construcción física de la nueva interfaz de comunicación CAN.

4.4.10 Consideraciones en la programación

Inicialmente se propuso una lógica para enviar múltiples mensajes en 2 canales con un solo código. Sin embargo, al cambiar el hardware del robot, ahora se poseen 4 canales de comunicación. El funcionamiento de los algoritmos de control para este tipo de sistemas establece una comunicación ya sea con cada una de las piernas o con cada una de sus articulaciones. Es decir, la información de cada una de las piernas se puede manejar individualmente. Debido a esto y a la arquitectura planteada, se desarrolla la ejecución de la comunicación en hilos de procesamiento separados, con el fin de disminuir el rendimiento por algún retraso en algunos de los canales que afecte al resto de ellos.

4.5 Diseño de la interfaz de pruebas

En esta sección se muestra el diseño de la interfaz de pruebas, además de consideraciones adicionales que surgen a raíz del proceso. Por último, se muestra la construcción física del prototipo.

4.5.1 Generación de conceptos de la interfaz de pruebas

La interfaz de pruebas brinda una respuesta rápida y segura con las principales funciones de un simulador, en donde se puede reproducir, pausar o reiniciar el ambiente virtual a petición del usuario. En Fig. 4.27 se pueden observar los conceptos propuestos, los cuales han sido divididos en dos categorías, alámbricos e inalámbricos.

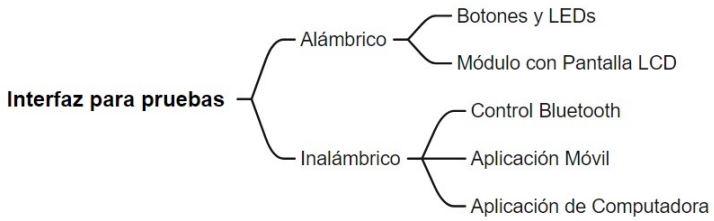


Figura 4.27: Generación de conceptos para la Interfaz de pruebas. Elaboración propia: Xmind

4.5.2 Conceptos de la interfaz de pruebas

Posteriormente, se generan los conceptos a partir de las ideas presentadas en el mapa mental ubicado en Fig. 4.27. A partir de ello se plantean los bocetos de las soluciones. En Fig. 4.28.a se plantea una solución que posee únicamente 4 botones para interactuar con la computadora central, la cual está conectada de manera física a sus puertos GPIO.

Por otro lado, en Fig. 4.28.b se propone emplear un control inalámbrico comunicado por medio de bluetooth para enviar los comandos al computador. En 4.28.c se plantea desarrollar una aplicación para una computadora individual, en la cual se tendrá una interfaz para observar datos de interés, así como ejecutar los comandos principales, la misma se conectaría por medio de Wifi al computador.

De manera similar, se plantea abordar la solución en 4.28.e, en donde se posee la idea planteada anteriormente, pero en una aplicación para teléfono. Por último, se presenta una interfaz de usuario con una pantalla en donde se puede ver el estatus del robot en 4.28.d, así como botones físicos para navegar a través de la interfaz y ejecutar los comandos más importantes, la misma estaría conectada de manera directa con el computador mediante un cable que permita la comunicación.

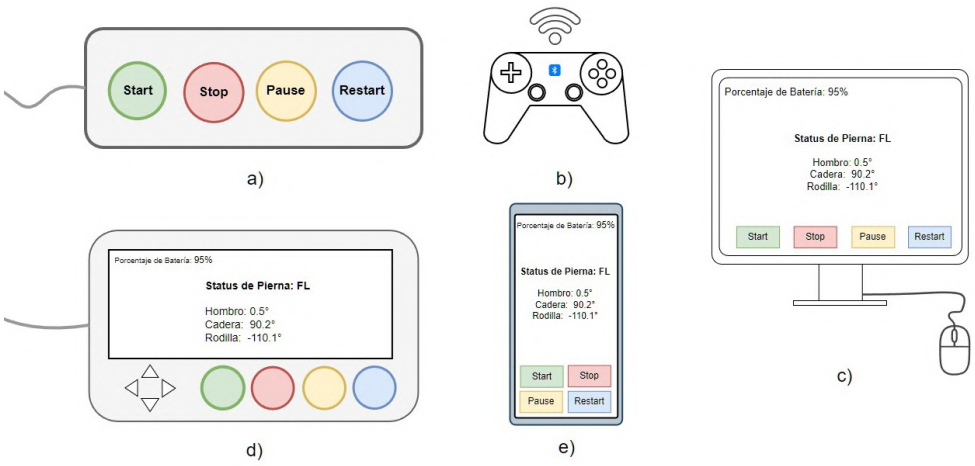


Figura 4.28: Bocetos para la interfaz de pruebas. a) Botones Físicos, b) Control Bluetooth, c) Aplicación de Computadora, d) Módulo con Pantalla y botones y e) Aplicación de teléfono inteligente. Elaboración propia: Diagrams.net

4.5.3 Selección de los conceptos de la Interfaz de pruebas

En el filtrado inicial, mostrado en la tabla 4.6, se indica que el módulo con pantalla no corresponde a un candidato óptimo en comparación con los demás. De sus puntos negativos se pueden rescatar la dificultad para realizarle modificaciones, el tiempo de desarrollo necesario y su costo. Sin embargo, se puede destacar la facilidad que le brinda al usuario para interactuar con el robot. Es por ello que se decide combinar la propuesta de un módulo con un aspecto visual con otra de las propuestas. La otra propuesta seleccionada para dicha combinación corresponde a la que posee únicamente botones, debido a que constituye del mismo tipo (alámbrica) y además, posee deficiencias en la facilidad de interacción.

Criterios de selección	**Conceptos**				
	Botones	Pantalla	Bluetooth	App Móvil	App PC
Facilidad de interacción	-	+	+	+	0
Seguridad	+	+	-	-	0
Modularidad	+	+	+	-	-
Facilidad de modificación	-	-	-	+	+
Consumo de energía	+	-	+	+	+
Costo	0	-	-	+	+
Tiempo de desarrollo	+	-	+	-	-
Suma +	4	2	4	5	4
Suma 0	1	0	0	0	1
Suma -	2	4	3	2	1
Evaluación Neta	2	-1	1	3	3
Lugar	2	4	3	1	1
¿Continuar?	Combinar	Combinar	Sí	Sí	Sí

Tabla 4.6: Matriz de Filtrado para los conceptos de la Interfaz de pruebas.

El módulo con pantalla ya poseía botones y una posible combinación con el boceto de solo botones obtendría un resultado muy similar. Es por ello que, en lugar de implementar una pantalla, la cual resulta costosa y requiere de un mayor tiempo para implementarla, se emplearán LEDs que indiquen el estado del robot, de manera que se mejore la interacción con el usuario sin impactar negativamente los otros criterios de selección. El resultado del nuevo concepto se puede ilustrar con el boceto presentado en Fig. 4.29, en donde se añadieron 3 LEDs en la parte inferior para indicar el estado principal del robot.

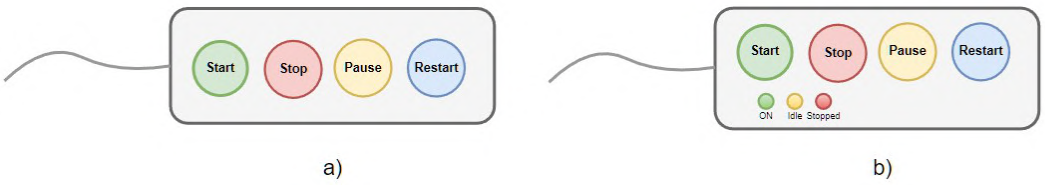


Figura 4.29: Combinación de Conceptos, a) Boceto de botones inicial b) Boceto de botones con LEDs Elaboración propia: Diagrams.net

En base a esto, se desarrolla la matriz de selección, ahora empleando el nuevo concepto en lugar de los dos iniciales. En la tabla 4.7 se observa los resultados de la selección, en donde se aprecia que los candidatos obtuvieron calificaciones relativamente similares. Si bien el concepto de los botones con los LEDs sigue siendo inferior en la facilidad de interacción con respecto a los demás, dicha brecha se mantiene pequeña. Por otro lado, este concepto resultado superior en el aspecto de seguridad debido a su baja latencia con el computador, y además, el tiempo de desarrollo es relativamente bajo, por lo cual se podrá entregar una versión robusta al cliente. Debido a esto, se decide continuar con esta propuesta en el proceso de desarrollo.

Criterios de selección	**Peso (%)**	**Conceptos**							
		Botones 2		Bluetooth		App Móvil		App PC	
		C.	E.	C.	E	C.	E	C.	E
Facilidad de interacción	16.1	3	0.48	4	0.65	4	0.65	4	0.65
Seguridad	12.9	5	0.65	3	0.39	3	0.39	4	0.52
Modularidad	12.9	4	0.52	4	0.52	3	0.39	3	0.39
Facilidad de Modificación	16.1	4	0.65	3	0.48	4	0.65	4	0.65
Consumo de energía	12.9	3	0.39	3	0.39	3	0.39	3	0.39
Costo	16.1	3	0.48	2	0.32	4	0.65	4	0.65
Tiempo de desarrollo	12.9	4	0.52	4	0.52	2	0.26	2	0.26
Total de puntos	100	26	3.7	23	3.3	23	3.4	24	3.5
Lugar		1		4		3		2	
¿Continuar?		**Sí**		No		No		No	

Tabla 4.7: Matriz de selección para la interfaz de pruebas

4.5.4 Desarrollo de la interfaz de pruebas

Para este problema se iniciará seleccionando los componentes, luego se mostrarán los pasos realizados para diseñar el circuito empleado y después se mostrará la implementación física final.

Selección de componentes

Como se mencionó anteriormente, el concepto ganador consiste de una interfaz de control que posee botones y LEDs. A continuación, se presentan una serie de consideraciones para establecer los criterios de diseño de este elemento.

- Será el controlador principal del algoritmo de control
- Mediante este, se detendrá el robot en caso de entrar en un estado inestable.
- Consiste el indicador principal del estado del robot

A partir de ello, se proponen los siguientes criterios de diseño.

- Los botones deben tener un tamaño considerable para ser actuados fácilmente en caso de una emergencia.
- Los indicadores LEDs deberán ser lo suficientemente notables como para observarse desde una distancia considerable.
- Las conexiones deben permitir un diseño modular.
- Debe ser capaz de detener el robot en cualquier situación.

A partir de ello, primeramente se seleccionan botones de un tamaño considerable, los cuales corresponden a botones cóncavos de 35mm de la marca Sparkfun. Estos poseen una durabilidad considerable y una respuesta rápida ante las pulsaciones, lo cual lo hacen un candidato ideal en otras aplicaciones como en los controles de videojuegos. En Fig. 4.30 se puede apreciar uno de los botones empleados, en donde cabe recalcar que otra ventaja de este producto es la variedad de colores en los que están disponibles. En el boceto inicial se presentaron 4 botones, correspondientes a las funciones de parar, iniciar, pausar y reiniciar, sin embargo, también se decide añadir una función extra con el fin de prever algún requerimiento adicional en un algoritmo en el futuro, o simplemente por motivos de conveniencia para facilitar la depuración de algún aspecto.

Por otro lado, se seleccionaron LEDs opacos de 10mm, lo cual los hace lo suficientemente notable en la interfaz y a cierta distancia de la misma. Estos también son de la marca Sparkfun y vienen en distintos colores, los mismos se presentan en Fig. 4.31

Figura 4.30: Botón Cóncavo seleccionado. Fuente: Sparkfun

Figura 4.31: LEDs de 10mm de diferentes colores. Fuente: Sparkfun

Diseño del circuito

El diseño del circuito de los botones se basa en el uso de las resistencias internas pull-up de la computadora central, las cuales permiten establecer una lectura por defecto de V_{cc}, de esta manera, los botones deberán llevar esta tensión hasta 0V una vez se activen. Con base en esto, se determina el uso de una lógica negativa, es decir, una activación de un botón corresponde a cambiar el voltaje de entrada de V_{cc} a 0 V, por lo cual se deberá detectar los flancos de caída en la lógica del controlador. En Fig. 4.32 se muestra el circuito propuesto para los 5 botones, en donde se tiene un nodo común conectado a GND y su otra terminal se conecta directamente al computador, sin necesidad de añadir componentes adicionales gracias al uso de las resistencias internas mencionadas anteriormente.

Por otro lado, para el circuito de los LEDs se deben tomar en cuenta sus características eléctricas. Tal como establece el fabricante, dichos elementos requieren de una corriente al menos $20mA$ para alcanzar el brillo requerido. Además, la tensión necesaria va desde 2V hasta 3.4V, lo cual depende del color del mismo. Los pines de la Raspberry Pi 4 son capaces de proveer señales de lógica de hasta 3.3V y una corriente de 16 mA [32]. A

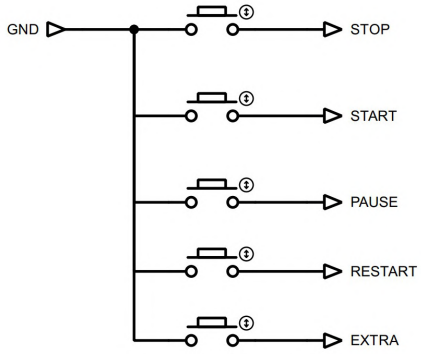


Figura 4.32: Circuito de los botones. Elaboración propia: Proteus

partir de ello, es evidente que los mismos no son capaces de iluminar los LEDs como lo establece el fabricante. Es por ello que se presenta un circuito empleando un transistor como el conversor de la lógica, el cual permite una fuente de alimentación de 5V para el circuito de los LEDs, tal como se aprecia en Fig. 4.33. En este se añadió la resistencia R_1 para regular la corriente de activación del transistor y R_2 para regular la tensión que se le provee al LED.

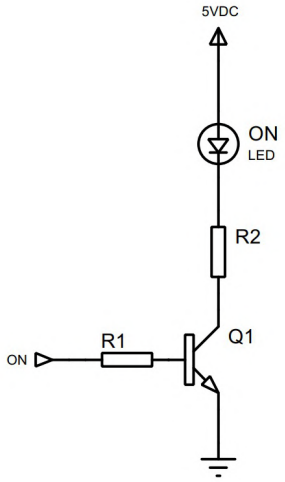


Figura 4.33: Circuito para la activación de los LEDs. Elaboración propia: Proteus

En este caso se emplean transistores 2N3704 debido a que coinciden con el tipo requerido, transistores bipolares BJT, y se tienen disponibilidad en el laboratorio donde se desarrolla el proyecto. Para dimensionar las dos resistencias, primero es necesario calcular la corriente mínima de saturación para que el transistor funcione como un interruptor, tal como se observa en la ecuación 4.8, recuperada de [17, 183].

$$I_{B(\min)} = \frac{I_{C(\text{sat})}}{\beta_{CD}} = \frac{20mA}{100} = 200\mu A \tag{4.8}$$

Posteriormente es necesario determinar la tensión a través de R_1, como se expresa en la

ecuación 4.9.

$$V_{R_1} = V_{ENT} - V_{BE} = 3.3V - 0.7V = 2.6V \tag{4.9}$$

De manera que se pueda estimar la resistencia máxima que se puede emplear en esta configuración, correspondiente a 13kΩ, tal como lo explica la ecuación 4.10.

$$R1_{\text{máx}} = \frac{V_{R1}}{I_{B(\text{min})}} = \frac{2.6\,\text{V}}{200\,\mu\text{A}} = 13\,\text{k}\Omega \tag{4.10}$$

En la ecuación 4.11 se presenta el cálculo de la tensión que pasará a través de los elementos LED y R_2, excluyendo al transistor.

$$V_C = V_{CC} - V_{CE(sat)} = 5V - 0.6V = 4.4V \tag{4.11}$$

Por otro lado, el cálculo de R_2 depende del color que emite el LED, debido a que su tensión de activación cambia. Los LEDs azules y verdes necesitan de 3.2 V mientras que los rojos tienen una tensión de activación de 2 V. Con base en esto, se calcula la tensión para los LEDs rojos y los azules y verdes, en las ecuaciones 4.12 y 4.13, respectivamente.

$$R_{2_{ROJO}} = \frac{V_C - V_{LED}}{I_C} = \frac{4.4V - 2V}{20mA} = 120\,\Omega \tag{4.12}$$

$$R_{2_{AZUL}} = \frac{V_C - V_{LED}}{I_C} = \frac{4.4V - 3.2V}{20mA} = 60\,\Omega \tag{4.13}$$

De esta manera, se completa la selección de componentes para el circuito de los LEDs, tal como se presenta en Fig. 4.34.

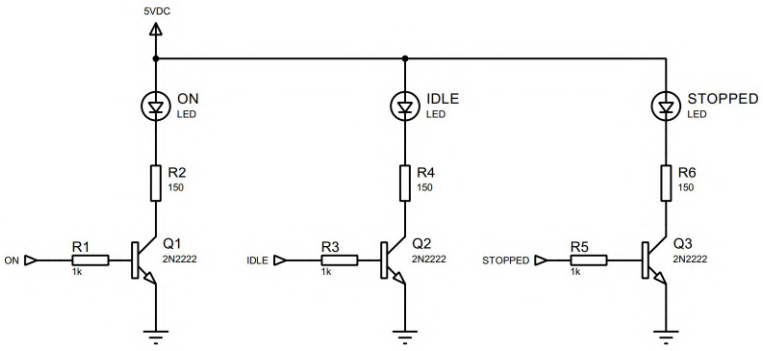


Figura 4.34: Circuito Final de los LEDs. Elaboración propia: Proteus

Construcción del prototipo

Para construir el modelo físico de la interfaz de control, primeramente se elaboró una carcasa en SolidWorks, tal como se ve en Fig. 4.35. Las consideraciones de diseño en esta etapa simplemente requerían de ajustar los componentes en una caja de un tamaño

Figura 4.35: Renderizado de la Interfaz de pruebas. Elaboración propia: Solidworks

razonable, pero a su vez, su tamaño permite que esta pueda ser cargada con una sola mano y accionar los botones de mayor importancia en caso de ser necesario.

Dicho prototipo se construyó mediante impresión 3D. Se empleó filamento de plástico PLA, y un relleno de 50%, debido a que las paredes son relativamente delgadas, por lo cual requieren de cierto nivel de rigidez. Por otro lado, también es importante que esta no sea excesivamente pesada, pero a su vez, al ser muy ligera puede dar una sensación de poca seguridad a la hora de sujetarlo. Los circuitos diseñados fueron construidos en una placa perforada, en donde se soldaron los componentes y cables necesarios. En Fig. 4.36 se muestra el interior del modelo final, en donde se observan las conexiones realizadas y la manera en que cada componente está sujetado adecuadamente a la carcasa. Asimismo, en Fig. 4.37 se presenta el exterior del prototipo, en el cual se aprecia tanto los botones y LEDs disponibles, como su respectiva etiqueta mediante el relieve en la superficie.

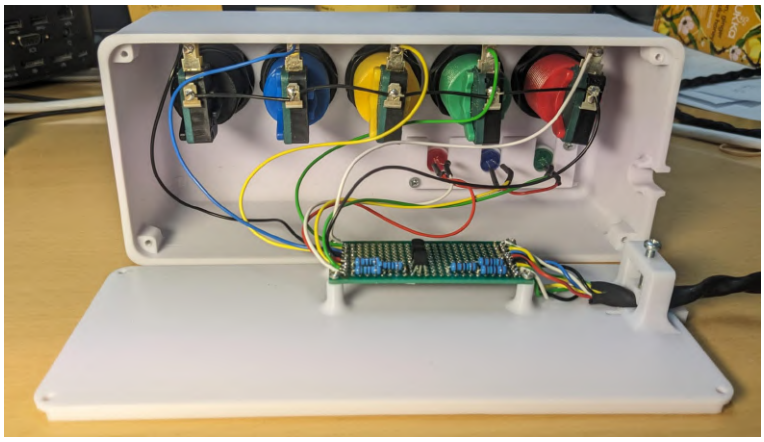


Figura 4.36: Interior del modelo físico de la interfaz de pruebas.

Figura 4.37: Exterior del modelo físico de la interfaz de pruebas.

Paro de emergencia

En la sección anterior se presentó el desarrollo de una interfaz que permite enviar señales lógicas al computador. En caso de querer detener al robot, la computadora debería leer la activación del botón "Stop" y ejecutar su protocolo para detener a los motores. Sin embargo, se decide tomar precauciones adicionales a este paro del robot, ya que detener el robot de manera lógica requiere de:

- Conservación de la integridad física de las conexiones de la red de comunicación
- Un código de bajo nivel sin ningún tipo de error
- Una interfaz de hardware que no sufra ningún tipo de complicaciones de búferes.
- La computadora central corre continuamente el programa sin retrasos.
- Una alimentación suficientemente estable en la computadora central y la red de comunicación.

Si bien se puede desarrollar un equipo que logre abarcar cada uno de estos aspectos, la naturaleza de este proyecto consiste en cambiar y probar nuevos códigos de bajo nivel al igual que competentes físicos, por lo cual, las probabilidades de que alguna de esas situaciones se dé al querer implementar una nueva función son muy altas. Los riesgos de no poder detener el robot con los comandos van desde dañar la integridad física de la estructura, daño de los motores, daño de la batería o incluso daño físico hacia los operarios. Esto debido a que los actuadores pueden funcionar de manera independiente de la computadora central, por lo cual, si el último comando enviado consiste en ejercer un torque muy alto, el motor lo seguirá haciendo hasta que reciba un comando que le indique lo contrario.

Es por ello que,, en este proyecto también se propone un paro de emergencia de un nivel más bajo, el cual consiste en cortar el suministro de electricidad a todo el robot. De esta manera se tendrá un respaldo ante cualquiera de los accidentes mencionados o incluso, una actuación rápida del operario cuando este quiere realizar alguna labor de mantenimiento

mientras el robot esté encendido. Por otro lado, tampoco requiere de tener la interfaz de pruebas conectada, por lo cual, el mismo podrá ser empleado cuando se desarrolle por completo las pruebas del sistema de locomoción y se emplean otras interfaces usuario-robot.

Debido al alto consumo de corriente de los actuadores, se decide emplear un relé, el cual será accionado por un interruptor de inicio y el botón de paro de emergencia será capaz de interrumpir dicha actuación. Este circuito se muestra en Fig. 4.38

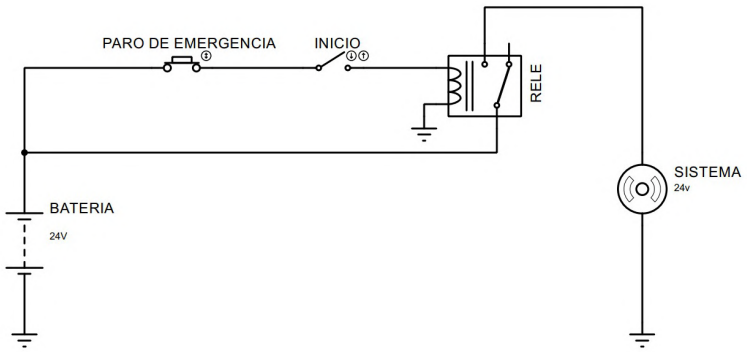


Figura 4.38: Circuito de paro de emergencia. Elaboración propia: Proteus

Dimensionamiento del Relé

Para dimensionar el relé necesario para el circuito, se debe tomar en cuenta, principalmente, el consumo máximo de los actuadores. Si bien la corriente de freno correspondería al máximo teórico del consumo, es poco probable que se dé la situación en donde los 12 actuadores entren en freno, debido a que al ejercer una fuerza estática sobre el robot, el torque de los actuadores se distribuirá de acuerdo con el estado actual de las articulaciones, tal como lo explica el jacobiano. Por otro lado, resulta de especial interés tomar en cuenta el consumo máximo cuando el robot ejecuta una secuencia de caminata rápida, ya que en ella se darán fuerzas dinámicas y múltiples impactos, por lo cual los torques obtienen valores significativos.

En base a esto, y a las simulaciones previas de la secuencia de caminado de trote mediante el control VMC, se determina el consumo máximo de corriente por pierna, presentado en la tabla 4.8, dicho consumo representa valores conservadores del torque promedio durante las situaciones más demandantes. Al tomar en cuenta la totalidad de los actuadores y los demás componentes eléctricos que consisten en el principal consumo, se determina la corriente máxima que se le debe suplir al robot para que este funcione en las condiciones más adversas, esto se observa en la tabla 4.9.

En base a dicha corriente, ahora se determina que el Relé deberá ser capaz de conducir al menos 110.3 A de manera continua. Por otro lado, es importante tomar en cuenta otros requerimientos como el voltaje de la carga o de la bobina de actuación. Debido a que el robot emplea una batería de 22.5V, se decide buscar un relé de 24V.

Articulación	Torque (Nm)	Consumo (A)
Hombro	15	7.17
Cadera	15	7.17
Rodilla	25	11.96
Total		26.3

Tabla 4.8: Consumo máximo de las articulaciones.

Componente	Consumo (A)
Piernas (4)	105.3
Computadora Central	3
Computadora de Percepción	2
Total	110.3

Tabla 4.9: Corriente de consumo máxima del robot

A partir de ello se selecciona el relé Digikey 1393315-9, mostrado en Fig.4.39, el cual puede conducir hasta 130A en corriente continua a 85°C, además es resistente a las vibraciones, lo cual es ideal para esta aplicación debido a los movimientos que se pueden dar en el robot. Cabe recalcar que el voltaje de activación mínimo de la bobina es de 14.4V, por lo cual no habrá problema en usar la batería de 22.5V.

Figura 4.39: Relé Seleccionado. Fuente: Digikey

4.6 Diseño de la detección del contacto

En esta sección se presentan los pasos desarrollados para la implementación física de la detección del contacto con el suelo del robot, los cuales abarcan desde las propuestas de diseño hasta la medición de las fuerzas aplicadas a las piernas en el estado de contacto.

4.6.1 Generación de conceptos de la detección del contacto

Para el caso de la detección del contacto se proponen distintos sensores que puedan llegar a detectar la situación requerida, esto se puede ver en Fig. 4.40. Estos se dividen en mediciones directas, las cuales consisten en colocar el sensor en la punta de las patas robóticas y que exista un contacto físico entre el sensor y el suelo. Por otro lado, también se presenta una medición indirecta, la cual consiste en la medición del torque de las articulaciones de la pata.

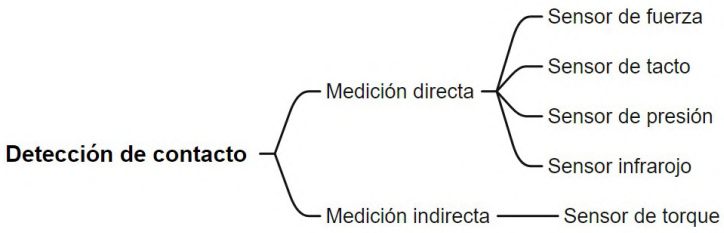


Figura 4.40: Generación de conceptos para la Detección del Contacto. Elaboración propia: Xmind

4.6.2 Selección de los conceptos de la detección del contacto

El proceso de la detección del contacto consistió en determinar el tipo de sensor más apropiado para la aplicación. En la tabla 4.10 se presenta el filtrado de las opciones propuestas, en donde se muestra que ninguna de las opciones obtuvo una calificación extremadamente baja, como sucedió en los dos casos anteriores. A partir de ello se toma la decisión de continuar el proceso de selección con las 4 opciones propuestas desde un inicio.

Criterios de selección	**Conceptos**			
	Fuerza	Tacto	Infrarrojo	Torque
Carga de Procesamiento	+	+	+	-
Facilidad de Implementación	-	-	-	+
Facilidad de modificación	+	+	0	+
Costo	-	0	0	+
Tiempo de desarrollo	-	-	-	+
Suma +	2	2	1	4
Suma 0	0	1	2	0
Suma -	3	2	2	1
Evaluación Neta	-1	0	-1	3
Lugar	3	2	3	1
¿Continuar?	Sí	Sí	Sí	Sí

Tabla 4.10: Matriz de Filtrado para los conceptos de la detección del contacto.

La matriz de decisiones para la selección de este problema muestra que el sensor de torque resulta superior a los demás, especialmente en los aspectos de la capacidad de detección, facilidad de modificación, costo y tiempo de desarrollo, tal como se observa en la tabla 4.11. En este proyecto, dicho candidato obtiene muchos puntos a favor debido a que emplea un sensor que se encuentra actualmente dentro de los actuadores, por lo cual su implementación se vuelve muy simple y además, no requiere de una componente mecánica y física, como lo podría requerir instalar un sensor de fuerza en la punta de las piernas.

Criterios de selección	**Peso (%)**	**Conceptos**							
		Fuerza		Tacto		Infrarrojo		Torque	
		C.	E.	C.	E	C.	E	C.	E
Carga de Procesamiento	15	3	0.45	4	0.39	4	0.60	2	0.30
Capacidad de detección	15	4	0.60	3	0.29	3	0.45	4	0.60
Facilidad de modificación	25	2	0.50	2	0.32	2	0.50	3	0.75
Costo	20	2	0.40	3	0.39	3	0.60	5	1.00
Tiempo de desarrollo	25	3	0.75	3	0.48	3	0.75	4	1.00
Total de puntos	100	14	2.7	15	2.9	15	2.9	18	3.7
Lugar		3		2		2		1	
¿Continuar?		No		No		No		**Sí**	

Tabla 4.11: Matriz de selección para la detección del contacto

4.6.3 Desarrollo de la detección del contacto

La trayectoria que sigue la pierna del robot supone que el punto inicial y final están en la misma altura, sin embargo, al encontrarse con obstáculos o imperfecciones en el terreno, esto ocasionará que el robot se desequilibre bruscamente en caso de interrumpir su trayectoria o en caso contrario, no tocar el suelo. Por otro lado, al cambiar los pasos de secuencia de caminata, es necesario asegurarse que las piernas de soporte en verdad están en contacto con la superficie, con el fin de garantizar la estabilidad del robot. Es por ello que el detectar el contacto juega un gran papel en el desempeño de la locomoción.

Existen múltiples métodos para solucionar este problema, sin embargo, algunos de ellos exigen una complejidad sumamente alta para el alcance de este proyecto. Es por ello que se presentará una solución a partir de datos empíricos, la cual funciona óptimamente para este robot en específico.

Como se mencionó anteriormente, en el apartado de selección de conceptos, se emplearán las medidas de torque de los actuadores para solventar este problema. Para ello se debe entender que, al cambiar entre fases el comportamiento de los torques cambia drásticamente, debido a las diferencias entre soportar una carga estática y una dinámica. Es por ello que resulta de gran utilidad observar el antes y el después de estas mediciones cuando el robot ejecuta una de sus trayectorias.

Esto se puede ver en Fig. 4.41, en donde entre los segundos 1 y 2, se balancea la pierna,

mientras que en el resto del tiempo se encuentra soportando el peso del robot. En cada una de las mediciones se observa una diferencia notable entre las fases. En el torque del hombro (T1), este aumenta significativamente y después llega a ser casi nulo. El torque de la cadera (T2), disminuye drásticamente y luego cambia de signo al impactar con el suelo. Por último, el torque de la rodilla (T3) es el que presenta las diferencias más notables, ya que, en el comportamiento dinámico de la pierna, su carga es casi nula en comparación a la estática.

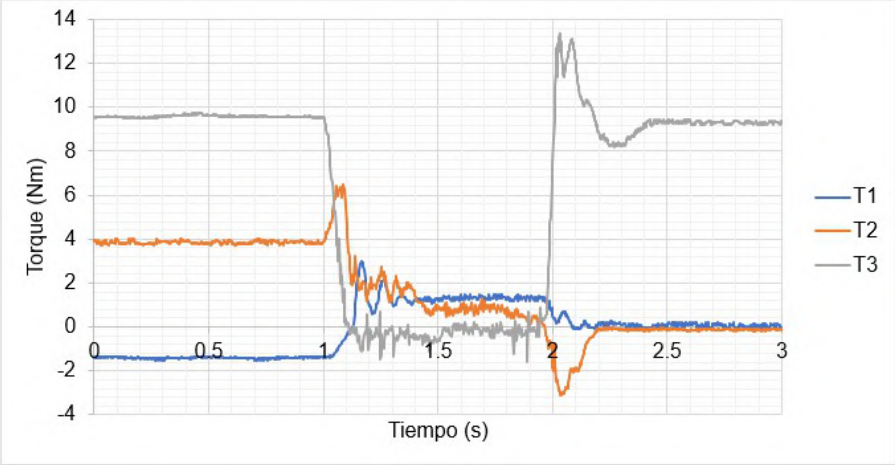


Figura 4.41: Lectura de los torques de los actuadores en el balanceo de una pierna. Elaboración Propia: Microsoft Excel

A partir de estas observaciones se podría desarrollar una detección de contacto que funcione únicamente con las mediciones de torque. Sin embargo, resulta útil recurrir a la estática de manipuladores, para observar el comportamiento teórico de los torques al mover el efector final de un punto a otro. Se emplea una fuerza equivalente a un cuarto del peso total del robot, es decir, alrededor de 37 N. En Fig. 4.42 se observa el diagrama de esta comparativa, en donde Fig. 4.42.a representa el inicio de una fase de balanceo y Fig. 4.42.b representa el final de esta. Los resultados del torque de los actuadores en estas dos situaciones se pueden observar en la tabla 4.12, en la cual se puede notar que el torque de la rodilla aumenta, mientras que el hombro permanece constante y la cadera aumenta considerablemente.

Estos cambios ocasionan que sea más difícil detectar el contacto en base a únicamente las medidas de los torques, ya que como se demostró anteriormente, al variar uno de los parámetros de la secuencia de caminata, las posiciones de las piernas serán diferentes, por lo cual los umbrales también cambiarán. Por otro lado, al detectar un obstáculo en durante la trayectoria, la pierna debería detenerse allí, por lo cual la posición final será diferente y también los torques en cada una de las articulaciones. Es decir, resulta muy complejo establecer directamente un umbral en el torque que contemple estas situaciones.

Es por ello que se decide emplear el cálculo de la fuerza vertical Fz, este mismo también corresponde a una de las componentes empleadas en un algoritmo más complejo de detección que posee el MIT Cheetah 3 [6]. Para ello, se deriva el cálculo de la fuerza a partir

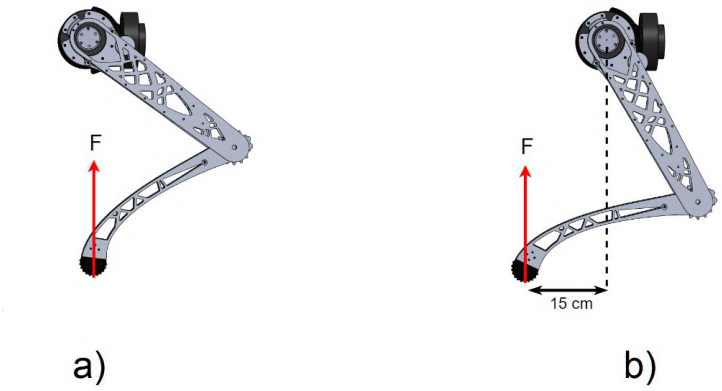


Figura 4.42: Reacción de la fuerza de contacto. a) Posicionamiento en X = 0.0 m, Z = -0.4 m. b) Posicionamiento en X = 0.15 m, Z = -0.4 m Elaboración Propia: Diagrams.net

Medida	Posición Inicial	Posición Final
Torque del hombro (Nm)	2.89	2.89
Torque de la cadera (Nm)	0.00	5.55
Torque de la rodilla (Nm)	8.61	10.41

Tabla 4.12: Comparación del torque ejercido de las articulaciones.

de la ecuación 2.10, resultando en la ecuación 4.14:

$$F = (J^T)^{-1}\tau \tag{4.14}$$

Debido a que solo se requiere de la fuerza en el eje Z, se puede simplificar el cálculo, como se presenta en la ecuación 4.15:

$$F_z = (J^T)^{-1}{}_{(3,0)} \cdot \tau_1 + (J^T)^{-1}{}_{(3,1)} \cdot \tau_2 + (J^T)^{-1}{}_{(3,2)} \cdot \tau_3 \tag{4.15}$$

De este modo se reduce ligeramente la carga computacional. En Fig. 4.43 se puede observar el cálculo de la fuerza vertical a partir de los torques mostrados en Fig. 4.41. En esta se observa que tiene un comportamiento similar al torque de la rodilla, sin embargo, ahora se tiene cierta certeza de que el umbral será similar en distintas situaciones.

Por otro lado, se observa un pequeño ruido en esta curva. Considerando que la detección depende si se sobrepasa el umbral o no, se decide reducir en parte este ruido. Esto se realiza mediante un filtro pasa bajos de primer orden. En Fig. 4.44 se puede ver la medida de la fuerza en función de diferentes valores de la constante del filtro, *alpha*. En esta detección se desea evitar un falso positivo debido al ruido, sin embargo, también se busca que se pueda detectar el contacto lo suficientemente rápido. En base a ello, se selecciona el valor de alpha de 0.5, debido a su rapidez similar al original y la manera en que se filtra la señal.

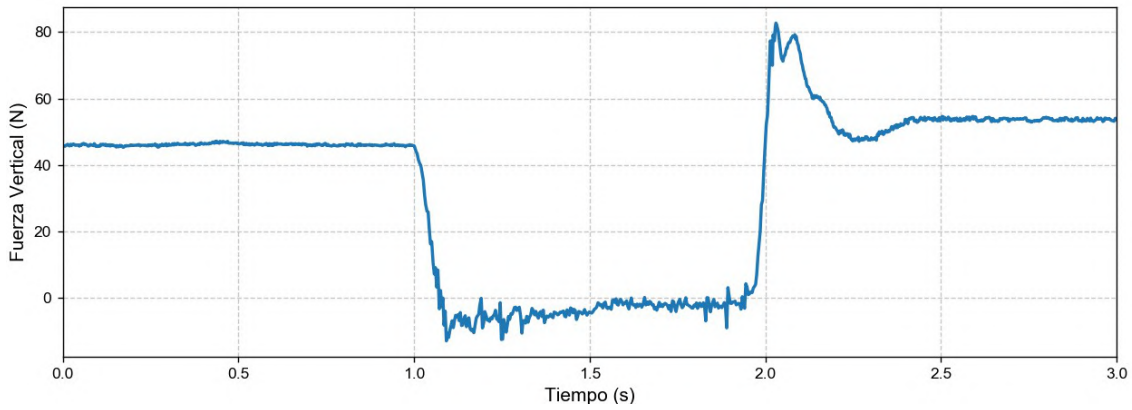


Figura 4.43: Cálculo de la fuerza vertical en una de las piernas durante el balanceo. Elaboración Propia: Python

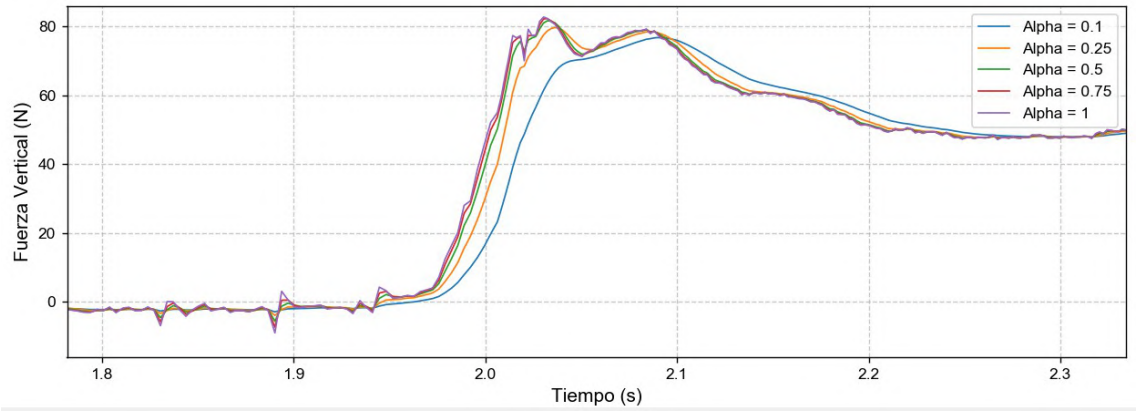


Figura 4.44: Filtrado del cálculo de la fuerza. Elaboración Propia: Python

De esta manera, se plantea la solución al problema de la detección de contacto, tal como se observa en el diagrama de la figura 4.45

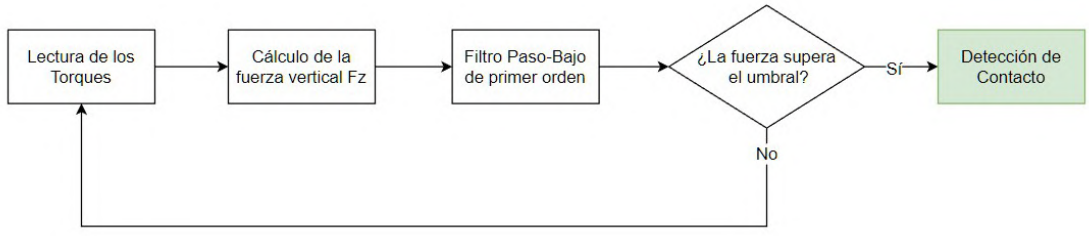


Figura 4.45: Diagrama de la solución propuesta para la detección de contacto. Elaboración Propia: Diagrams.net

Determinación del Umbral

Para determinar la fuerza vertical que cada una de las piernas deberá soportar cuando se dé el contacto, se espera que el peso del robot sea distribuido entre las 4 piernas. Por ello, dicha fuerza debería ser cercana a un cuarto del peso del robot. Si bien se posee

una medida teórica del peso, modificaciones posteriores en el diseño mecánico, además de la adición de nuevos componentes hacen que medir empíricamente este valor sea más confiable que usar únicamente el teórico.

Para ello, se envían los comandos respectivos al robot, de manera que este se posicione a 40cm sobre el suelo. A partir de ello, se miden los torques ejercidos por los motores, esto en 5 ocasiones. Posteriormente se calcula el promedio de estos torques para después calcular la fuerza percibida vista por los motores, la cual se presenta la tabla 4.13. Si bien la fuerza de impacto es mayor que la fuerza en estado de reposo, al alcanzar el valor de la fuerza de soporte, se tendrá certeza que la superficie contactada será lo suficientemente rígida como para soportar el peso del robot.

Articulación	**Torque Promedio (Nm)**
Hombro	-0.617
Cadera	-5.293
Rodilla	7.641
Fuerza vertical (N)	44.158

Tabla 4.13: Determinación de la Fuerza vertical presente en las patas del robot

4.7 Diseño de la secuencia de caminata

Esta sección muestra la aplicación de la metodología de diseño para seleccionar y desarrollar las componentes principales de la secuencia de caminata del robot. Asimismo, se presenta la validación mediante simulaciones, en conjunto con consideraciones adicionales como lo es la seguridad y la estimación de la posición del robot.

4.7.1 Generación de conceptos de la secuencia de caminata

Las secuencias de caminata propuestas para validar este trabajo se presentan en Fig. 4.46, en las cuales se subdividieron de acuerdo con su estabilidad estática. Esto hace referencia a si robot requiere de un movimiento continuo para mantener su equilibrio o si por el contrario, en todos los pasos de la secuencia este se encuentra estable. Por otro lado, los conceptos presentados en este problema son ritmos de marcha que ya han sido probados en el simulador previamente, esto debido a la complejidad que requiere adaptar uno completamente al modelo con el que se está trabajando.

Si bien la trayectoria de balanceo no es un elemento crucial en la secuencia de caminado en general, esta puede tomar importancia cuando se desea tener un rendimiento fluido y cuando se le presta atención al impacto con el suelo. En Fig. 4.47 se observan las 3 trayectorias propuestas, las cuales son usadas frecuentemente en otros robots cuadrúpedos hasta cierto punto.

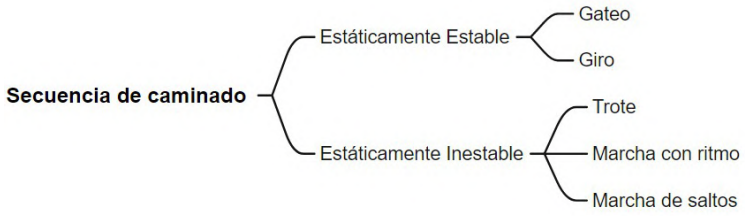


Figura 4.46: Generación de conceptos para la secuencia de caminado. Elaboración propia: Xmind

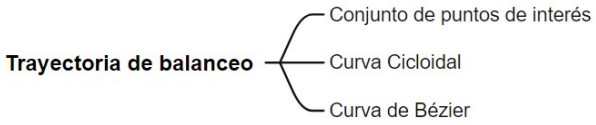


Figura 4.47: Generación de conceptos para la trayectoria del balanceo de la pata. Elaboración propia: Xmind

Por último, se presentan los distintos algoritmos de locomoción a usar, los cuales varían desde los más simples que consiste en el control posicional de las articulaciones, hasta el mayor grado de complejidad que se ha alcanzado en este robot con el control MPC. Esto se puede apreciar en Fig. 4.48, en el cual una locomoción al nivel de las patas hace referencia a que los torques de cada articulación se calculan a partir de una fuerza que debe ser ejercida por la totalidad del brazo. Por otro lado, una locomoción al nivel de las articulaciones no posee dicho cálculo de fuerza, ya que esta se centra únicamente en la posición o el torque deseado de cada uno de los actuadores.

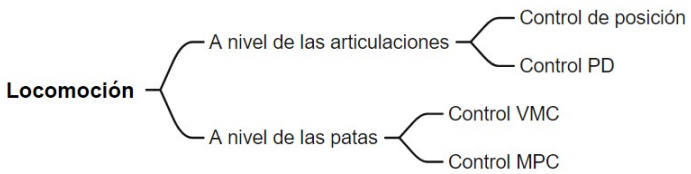


Figura 4.48: Generación de conceptos para el algoritmo de la Locomoción. Elaboración propia: Xmind

4.7.2 Conceptos de la secuencia de caminado y la locomoción

A continuación, se presenta la construcción de los conceptos para las secuencias de caminado del problema del procesamiento de la información.

Gateo con control posicional

El primero de ellos consiste en emplear la secuencia del gateo, en conjunto con una generación de trayectorias de la curva de Bézier y por último, el control más simple de

todos, el posicional, tal como se muestra en Fig. 4.49

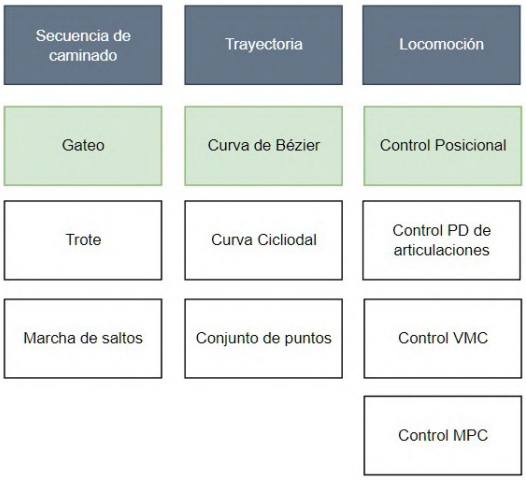


Figura 4.49: Construcción del Concepto "Gateo Posicional". Elaboración propia: Diagramas.net

Trote con control VMC

Otros de los conceptos propuestos, corresponde a emplear el algoritmo de control VMC, el cual es uno de los que poseen mejor desempeño sin requerir un alto poder computacional. Esto se implementa en conjunto con una trayectoria cicloidal y basándose en la secuencia de caminata de trote, esto se presenta en Fig. 4.50.

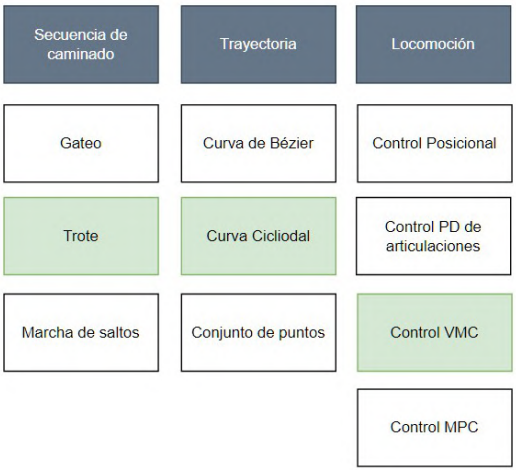


Figura 4.50: Construcción del Concepto "Trote VMC". Elaboración propia: Diagramas.net

Marcha de saltos con control MPC

En Fig. 4.51 se puede ver el tercer concepto para este problema, el cual consiste en usar el algoritmo de control MPC, correspondiente a uno de los más avanzados, esto en conjunto

con una trayectoria de balanceo de tipo cicloidal y empleando la marcha de saltos como la secuencia de caminado, esto se presenta en Fig. 4.51.

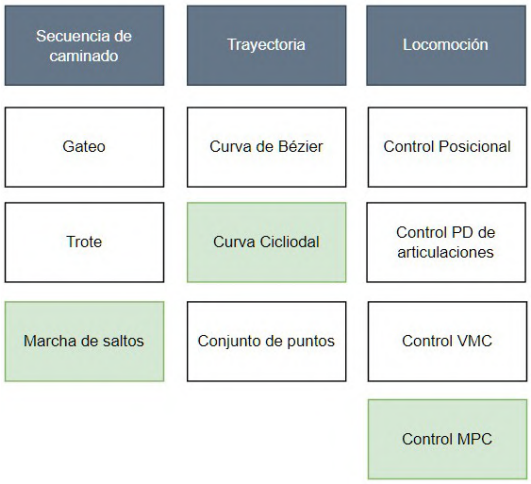


Figura 4.51: Construcción del Concepto "Saltos MPC". Elaboración propia: Diagramas.net

Gateo con control PD

Por último, se presenta un concepto que emplea la secuencia de caminata de gateo, nuevamente, pero ahora emplea un conjunto de puntos de interés como trayectoria y además, basa su locomoción en un control PD al nivel de las articulaciones, tal como se observa en Fig. 4.52

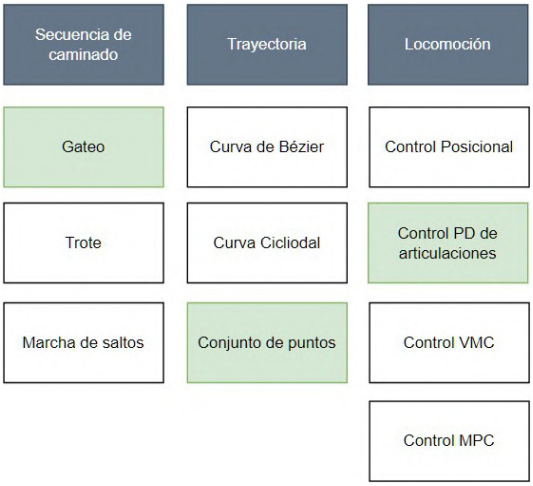


Figura 4.52: Construcción del Concepto "Gateo PD". Elaboración propia: Diagramas.net

4.7.3 Selección de los conceptos del procesamiento de información

El filtrado de conceptos en el problema del procesamiento de la información se muestra en la tabla 4.14, el mismo indicó que el candidato de la marcha de saltos por MPC resulta muy inferior a los demás. Las razones de esto consisten en una carga computacional alta, complejidad alta a la hora de implementarse físicamente y un tiempo de desarrollo relativamente alto. La principal causa de la calificación en estos aspectos corresponde al uso del algoritmo de control MPC. Es por ello que se podría realizar cambios al candidato, pero si no se cambia el algoritmo, el resultado será muy similar. Debido a ello, se decide no tomar en cuenta a este candidato para la etapa posterior.

Criterios de selección	**Conceptos**			
	Gateo Pos.	Trote VMC	Saltos MPC	Gateo PD
Frecuencia de Lazo de Control	+	0	-	+
Velocidad de movimiento	-	+	+	-
Consumo de energía	+	-	-	+
Estabilidad del cuerpo	+	+	-	+
Tiempo de desarrollo	+	-	-	0
Suma +	4	2	1	3
Suma 0	0	1	0	1
Suma -	1	2	4	1
Evaluación Neta	3	0	-3	2
Lugar	1	3	4	2
¿Continuar?	Sí	Sí	No	Sí

Tabla 4.14: Matriz de Filtrado para los conceptos del procesamiento de información

A la hora de aplicar la matriz de decisiones a los 3 candidatos restantes, se obtuvo la tabla 4.15, en donde se observa que el gateo por control posicional resultó el mejor candidato, seguido del gateo por control PD. Si bien la calificación final es relativamente similar, se decide continuar únicamente con el gateo por control posicional debido al bajo tiempo de implementación, el cuál será de vital importancia para realizar posibles iteraciones.

Criterios de selección	**Peso (%)**	**Conceptos**					
		Gateo Pos.		Trote VMC		Gateo PD	
		C.	E.	C.	E	C.	E
Frecuencia del lazo de Control	15	4	0.60	2	0.30	3	0.45
Velocidad de movimiento	20	3	0.60	5	1.00	3	0.60
Consumo de energía	20	5	1.00	3	0.60	5	1.00
Estabilidad del cuerpo	25	4	1.00	3	0.75	5	1.25
Tiempo de desarrollo	20	5	1.00	2	0.40	3	0.60
Total de puntos	100	21	4.2	15	3.1	19	3.9
Lugar		1		3		2	
¿Continuar?		**Sí**		No		No	

Tabla 4.15: Matriz de selección para la Secuencia de caminado

4.7.4 Desarrollo de la secuencia de caminata

El desarrollo de la secuencia de caminata consiste en primeramente desarrollar las curvas las fases de balanceo y de soporte, después desarrollar la lógica del movimiento de las piernas y por último, estructurar el control que se empleará en el robot.

Desarrollo de la trayectoria de balanceo

En el apéndice A.1.3 se presenta la curva de Bézier propuesta por el MIT, la cual desarrolla una trayectoria que permite un movimiento fluido y con pocas pérdidas de energías. Sin embargo, en ella se emplea una selección de puntos de control fija y, además, se basa en las dimensiones del MIT Cheetah 3, el cual es más grande que el robot que se usa en este proyecto. Es por ello que surge la necesidad de ajustar esta trayectoria al modelo empleado.

Para ello, se toman dos parámetros como referencia, correspondientes a la longitud del recorrido, L, y la altura de la trayectoria, H. En este caso se emplean únicamente dimensiones para situar los puntos de control de la curva, sin embargo, en otros casos resulta de especial interés emplear la velocidad en el eje horizontal como parámetro.

Primeramente, se puede observar que los puntos de control fueron diseñados para iniciar en las coordenadas (-200mm, 500mm), sin embargo, en este caso resulta más conveniente emplear el inicio en 0mm, 0mm y luego aplicar una compensación con el punto en donde se encuentra la pierna en ese momento. La parametrización de la curva se basa en buscar el factor de proporcionalidad entre el valor original el punto de control y el parámetro longitud de la curva, de igual manera con la altura. En base a esto, se obtiene la tabla 4.16, en donde se calcularon los nuevos puntos de control que se pueden adaptar a cualquier situación. Es importante recalcar que es posible parametrizar esta curva debido a que se puede mantener los puntos de velocidad y aceleración cero del diseño original y además, se mantiene la proporcionalidad en las tasas de movimiento de la pierna.

Punto	Eje X (mm)	Eje Y (mm)
c0	0	0
c1	$-\frac{L}{5}$	0
c2	$-\frac{L}{4}$	0
c3	$-\frac{L}{4}$	H
c4	$-\frac{L}{4}$	H
c5	$\frac{L}{2}$	H
c6	$\frac{L}{2}$	H
c7	$\frac{L}{2}$	$1.3 \cdot H$
c8	$1.25 \cdot L$	$1.3 \cdot H$
c9	$1.25 \cdot L$	$1.3 \cdot H$
c10	$1.2 \cdot L$	0
c11	L	0

Tabla 4.16: Puntos de control parametrizados para la curva de Bézier

A partir de los nuevos puntos de control obtenidos, es posible desarrollar una trayectoria de Bézier para la secuencia de caminata de este robot. Para ello, se emplea una longitud de 15cm y una altura de 10cm, dando como resultado la curva de Fig. 4.53. En ella se puede ver como la forma se asemeja a la trayectoria original. Por otro lado, al observar Fig. 4.54, se observa la velocidad del efector final y como en el eje vertical se mantiene en 0 en el punto inicial y el final, lo cual comprueba que la funcionalidad original se conservó al parametrizar la curva.

Asimismo, el control de posición empleado en los motores requiere de un comando de velocidad. Para ello, se realiza una diferenciación discreta de los comandos de ángulos de las articulaciones, de la manera en que se observa en 4.16.

$$v(k) = \frac{p(k) - p(k-1)}{dt} \tag{4.16}$$

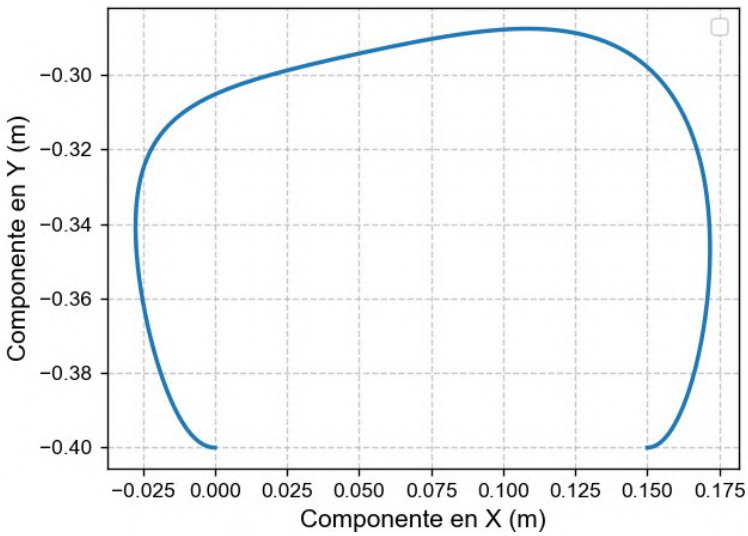


Figura 4.53: Posición del efector final en la trayectoria de Bézier obtenida. Elaboración propia: Python

Curva de las piernas de soporte

Las piernas que se encuentran en contacto con el suelo durante la secuencia de caminata deberán mover el centro de masa hacia adelante. Es decir, desde el punto de vista de la trama de las piernas individuales, se deberá mover el efector final hacia atrás. Esta trayectoria resulta relativamente simple en comparación con la de balanceo y se puede describir como se presenta en la ecuación 4.17.

$$p_x(t) = P_{x_0} - \frac{L}{3} \cdot t \tag{4.17}$$

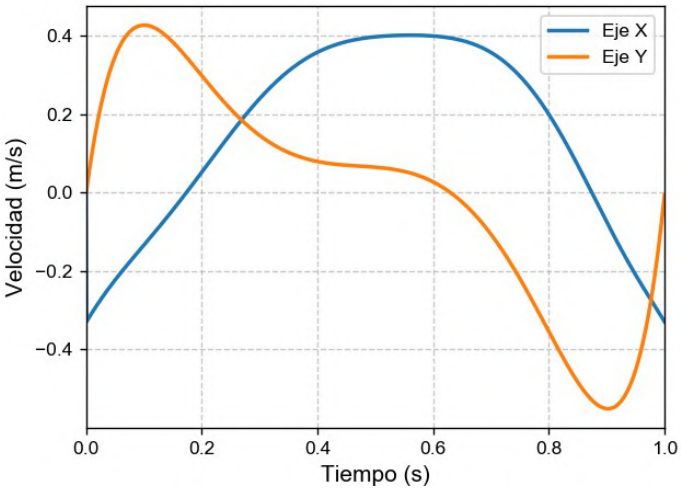


Figura 4.54: Velocidad del efector final en la trayectoria de Bézier obtenida. Elaboración propia: Python

4.7.5 Secuencia de caminata

Para desarrollar la lógica de la secuencia de caminata del gateo, se tomará como referencia el esquema previo que se ha desarrollado en trabajos anteriores. Este consiste de 4 pasos:

1. Mover el centro de masa hacia la derecha y balancear la pierna frontal izquierda
2. Balancear la pierna trasera izquierda
3. Mover el centro de masa hacia la izquierda y balancear la pierna frontal derecha
4. Balancear la pierna trasera derecha.

En base a la realimentación proporcionada por los estudiantes responsables de dicha secuencia de caminata, resultó imposible implementarla en el robot físico, ya que este tenía problemas de estabilidad. Con el fin de solventar dicho problema, se propone una secuencia de gateo más larga y más lenta, pero a cambio de dicha simplicidad, se busca asegurar la estabilidad del cuerpo del robot, de manera que posteriormente se pueda implementar exitosamente en el robot real.

En Fig. 4.55 se muestra la secuencia de gateo extendida, la cual consta de los siguientes pasos:

1. Mover las piernas a la posición inicial.
2. Mover el centro de masa hacia la derecha.
3. Balancear la pierna trasera izquierda.
4. Balancear la pierna delantera izquierda.

5. Mover el centro de masa hacia la izquierda.
6. Balancear la pierna trasera derecha.
7. Balancear la pierna delantera derecha.

Cabe recalcar la importancia del posicionamiento inicial de las piernas, ya que esto permite un funcionamiento cíclico y que cada balanceo de pierna inicie en el mismo punto. Esto simplifica la estabilidad del cuerpo, debido a que las condiciones de cada paso son similares, por lo que será fácil modificar todas de ser necesario. En la tabla 4.17 se muestra el posicionamiento relativo de cada pierna con respecto a sus tramas del hombro, en donde L representa la longitud del balanceo de pierna y W consiste en la distancia que se mueve el centro de masa hacia los lados.

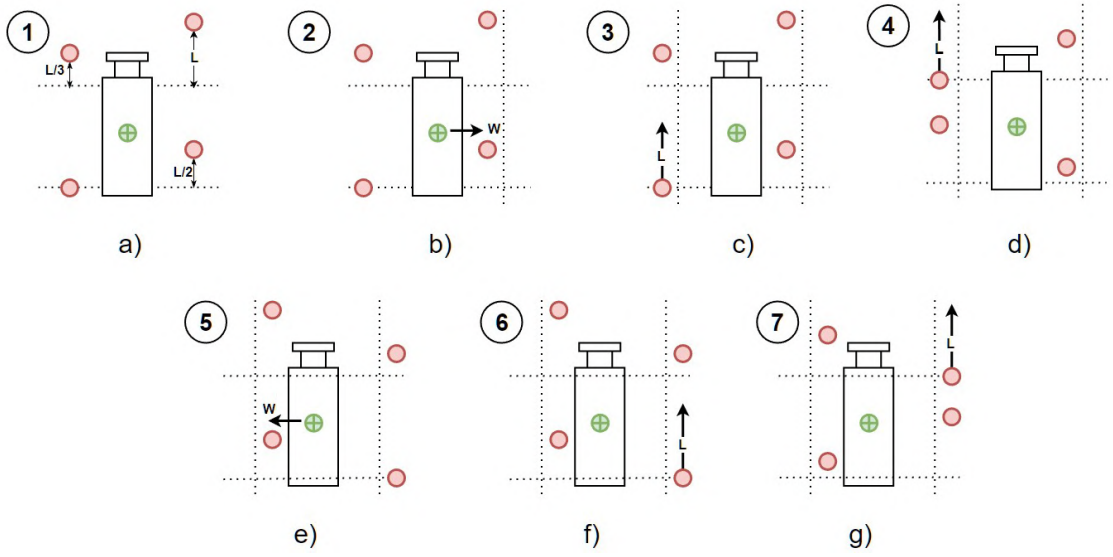


Figura 4.55: Secuencia de caminata. a) Posicionamiento Inicial. b) Movimiento del CoM hacia la derecha. c) Balanceo de la pierna trasera izquierda. d) Balanceo de la pierna frontal Izquierda. e) Movimiento del CoM hacia la izquierda. f) Balanceo de la pierna trasera derecha. g) Balanceo de la pierna frontal derecha. Elaboración propia: Diagrams.net

4.7.6 Estructura del Algoritmo de Control

El control seleccionado para esta secuencia de caminata corresponde al posicional. Debido a que los motores poseen un control interno de posición, se debe calcular los ángulos necesarios para cada uno de ellos y enviar los comandos respectivos.

En Fig. 4.56 se presenta el control propuesto. Este recibe inicialmente una posición cartesiana deseada, para después obtener los ángulos de las articulaciones mediante la cinemática inversa desarrollada en trabajos anteriores. Posteriormente, se verifica que los comandos estén dentro del espacio de trabajo del robot. Luego se adaptan estos comandos,

Paso	Piernas							
	FL		FR		BL		BR	
	Eje X	Eje Y	Eje X	Eje Y	Eje X	Eje Y	Eje X	Eje Y
1	$L/3$	0	L	0	0	0	$L/2$	0
2	$L/3$	W	L	$-W$	0	W	$L/2$	$-W$
3	0	W	$L/2$	$-W$	L	W	$L/3$	$-W$
4	L	W	$L/3$	$-W$	$L/2$	W	0	$-W$
5	L	$-W$	$L/3$	W	$L/2$	$-W$	0	W
6	$L/2$	$-W$	0	W	$L/3$	$-W$	L	W
7	$L/3$	$-W$	L	W	0	$-W$	$L/2$	W

Tabla 4.17: Posicionamiento relativo de las piernas del robot durante la secuencia de caminata de gateo.

los cuales fueron calculados en base a las coordenadas cinemáticas, a los actuadores reales, por lo que se debe de tener en cuenta reducciones por juego de engranes y cambios de dirección debido a la configuración de los mismos. Por último, se genera el comando para comunicarse con los actuadores.

A su vez, después de enviar este comando, el actuador responderá con la información referente a las posiciones, velocidades y torques del mismo. Por lo que deberá calcularse el ángulo multi-giro, como se mencionó anteriormente, y después adaptar dichas mediciones a las coordenadas cinemáticas, de manera que se obtiene el estado de la pierna en base a las mediciones.

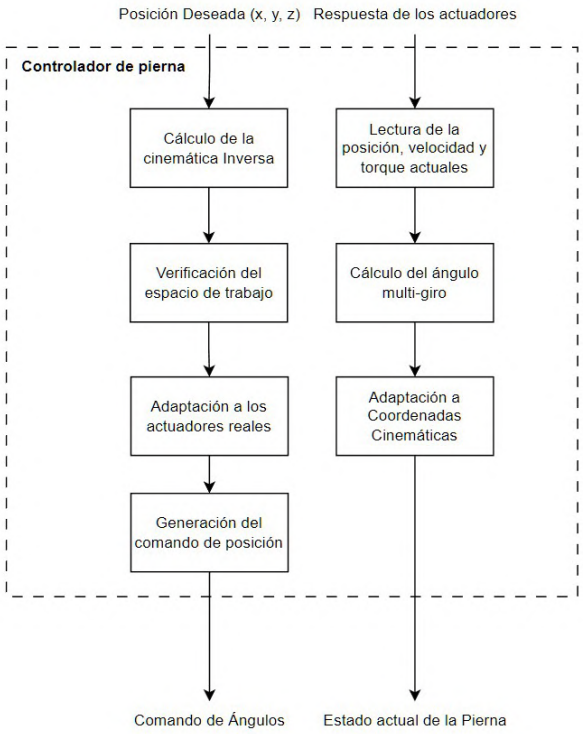


Figura 4.56: Diagrama del control empleado. Elaboración propia: Diagrams.net

Seguridad

Asimismo, se proponen funciones de seguridad las cuales garantizan que el robot se encuentre en condiciones óptimas para su funcionamiento, además de que el propio algoritmo envíe comandos válidos dentro de las capacidades del robot. En Fig. 4.57 se presentan los módulos para controles de torque y posición. Cabe destacar que incluso cuando se está realizando un control únicamente de posición es importante observar el torque ejercido por el actuador, debido a que este indica posibles impactos o atascamiento de las articulaciones. Al detectar estas situaciones del torque, se decide detener por completo el robot, es decir, las articulaciones no ejercerán fuerza, debido a que, si este solamente se pausa, aún estará en el estado de atascamiento.

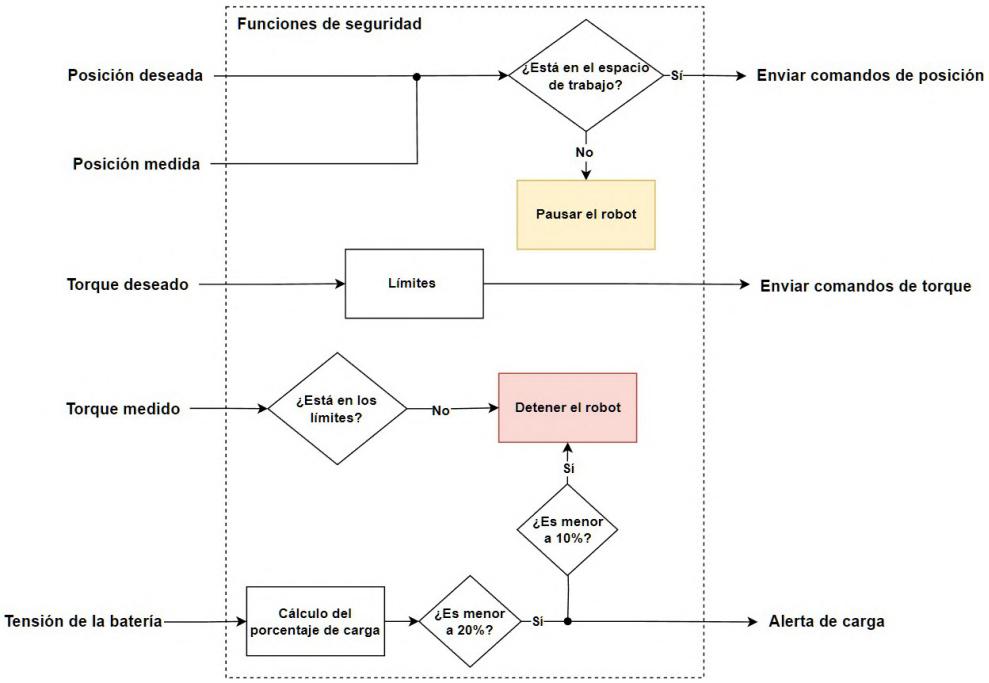


Figura 4.57: Funciones de seguridad empleadas en la arquitectura. Elaboración propia: Diagrams.net

Por otro lado, al implementar un control de posición, resulta posible que por errores de código o especificaciones del usuario, estos se salgan del espacio de trabajo, sin embargo, no es necesario detener por completo el robot, ya que involucra un posible impacto. En dichos casos, debido a que se tiene cierta certeza de que la situación es poco severa, se puede pausar el robot con un comando de posición, lo cual agiliza las pruebas. Asimismo, se decide realizar la misma acción en caso de que las mediciones de posición se encuentren fuera de las especificaciones, ya que únicamente el torque indicará cuando el estado del mismo pone en peligro la integridad del robot o su entorno.

Lo valores del campo de trabajo de cada una de las patas del robot se presentan en la ecuación 4.18, los mismos hacen referencia a de la pata izquierda frontal.

$$\text{Espacio de trabajo} : \begin{cases} \text{Hombro} : [0\ ^\circ,\ 15^\circ] \\ \text{Cadera} : [\text{-}90\ ^\circ,\ 90^\circ] \\ \text{Rodilla} : [\text{-}125\ ^\circ,\ 0^\circ] \end{cases} \tag{4.18}$$

Por otro lado, el valor del torque máximo del actuador es de 25 Nm, y al emplear la constante del mismo se obtiene que esto corresponde a una corriente de 11.96 A, de la manera en que se presenta en la ecuación 4.19.

$$\text{Corriente Límite} = \frac{25Nm}{2.09A/Nm} = 11.96A \tag{4.19}$$

Adicionalmente, se comprueba el porcentaje de carga de la batería, ya que su voltaje de operación nominal corresponde a 22.5 V, sin embargo, los motores pueden trabajar con hasta 12 V. Es decir, los actuadores pueden descargar las celdas de la batería más allá del límite establecido por el fabricante. Es por ello que se decide leer el voltaje de la misma, mediante un comando a los actuadores, y calcular el porcentaje de carga en base al apéndice A.2.1. De esta manera, se detiene el robot cuando la carga es sumamente baja y se emite un aviso en la consola.

Dichas funciones de seguridad resultan apropiadas para este caso, pero pueden ser simples hasta cierto punto. Este tema se puede extender a considerar el movimiento de las 4 piernas y determinar cuándo algunas de ellas se entrecruzan o incluso, mediante el uso de algoritmos de percepción del ambiente, se puede detectar la presencia de humanos y detener el avance del robot para garantizar un ambiente seguro.

4.7.7 Simulación de la secuencia de caminado

Para realizar una validación previa de la secuencia de caminata desarrollada, se emplea el software de simulación Webots. En Fig. 4.58 se muestra el balanceo de las cuatro patas. En Fig. 4.58.a se puede observar como el robot se desequilibra al tratar de levantar la pata trasera izquierda, de igual forma que ocurre con la trasera derecha en Fig. 4.58.c. Con las piernas delanteras, el robot no sufre de desbalances, tal como se ve en Fig. 4.58.b y Fig. 4.58.d.

Estos desbalances indican que el posicionamiento de las piernas no fue el adecuado. A partir de ello, se debe cambiar el posicionamiento inicial o la distancia en la que se mueve el centro de masa hacia los lados. En base a la experiencia con el robot real, valores superiores a $W = 0.075$ m ocasionan que el robot se pueda caer hacia los lados. Por lo tanto, se debe modificar la coordenada X. Para ello, se agregará una distancia de compensación.

Sin embargo, primeramente vale la pena analizar la estabilidad de la configuración de la secuencia. Para ello, se emplea el método de la proyección del centro de gravedad, el cual consiste en trazar polígono con las piernas que se encuentran en la fase de soporte,

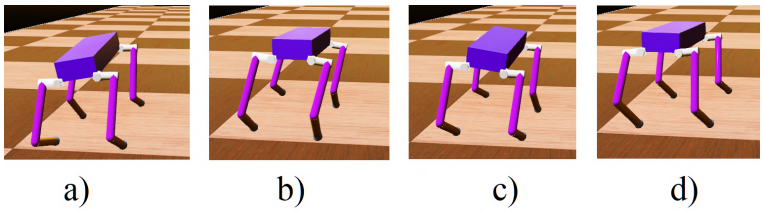


Figura 4.58: Simulación de la secuencia de caminata de gateo inestable. a) Balanceo de la pierna trasera izquierda. b) Balanceo de la pierna delantera izquierda. c) Balanceo de la pierna trasera derecha. d) Balanceo de la pierna delantera derecha. Elaboración propia: Webots

en este caso 3, y si el centro de gravedad se encuentra dentro de este polígono, existirá una estabilidad estática [18]. De esta manera, entre más cerca el COG del robot esté del centroide de este polígono, la estabilidad será mucho mejor.

En Fig. 4.59 se observan los polígonos de estabilidad en el momento en el que se va a balancear cada una de las 4 piernas. Aquí se observa que el centro de gravedad se encuentra dentro del polígono en todas las ocasiones, sin embargo, al balancear las piernas traseras, en Fig. 4.59.a y Fig. 4.59.c, este se encuentra muy cerca del límite, por lo cual existe un riesgo de inestabilidad a la hora de ejecutar la trayectoria de la pierna y mover el centro de masa hacia adelante.

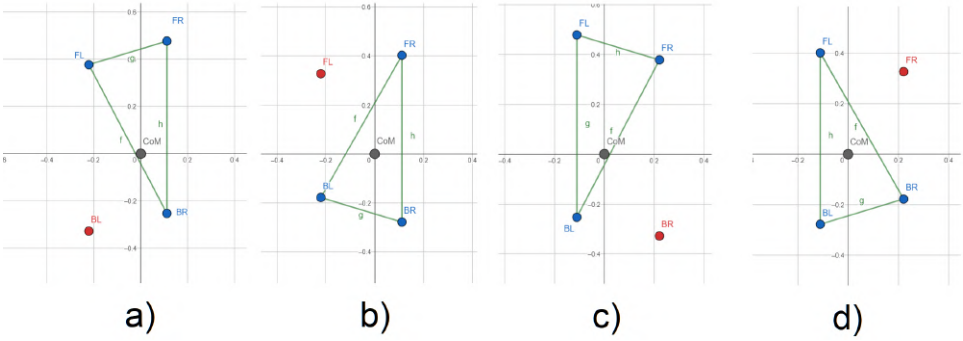


Figura 4.59: Análisis de estabilidad de la secuencia de caminado. a) Balanceo de la pierna trasera izquierda. b) Balanceo de la pierna delantera izquierda. c) Balanceo de la pierna trasera derecha. d) Balanceo de la pierna delantera derecha. Elaboración propia: GeoGebra

Para solventar esto, se introduce una compensación de 0.075m, la cual fue obtenida de manera iterativa hasta obtener un resultado óptimo. En Fig. 4.60 se puede observar como ahora el centro de gravedad se encuentra dentro del polígono de una manera similar en las 4 posibles situaciones. De esta forma, se encuentra un equilibrio en el que el robot podrá balancear las piernas traseras sin reducir notablemente la estabilidad al balancear las piernas delanteras.

Posteriormente se volvió a realizar la validación previa en el simulador con los ajustes mencionados. El resultado de ello se presenta en Fig. 4.61, en donde se encuentra que el los movimientos del robot son estables y no se nota la presencia de desequilibrios notables

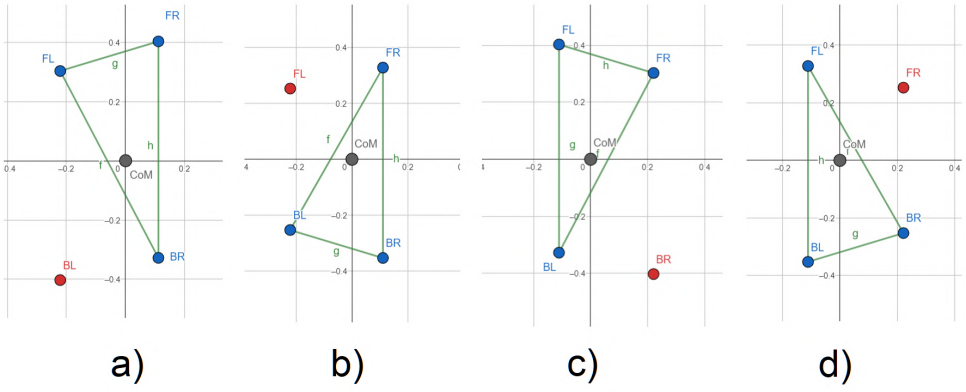


Figura 4.60: Análisis de estabilidad de la secuencia de caminado modificada. a) Balanceo de la pierna trasera izquierda. b) Balanceo de la pierna delantera izquierda. c) Balanceo de la pierna trasera derecha. d) Balanceo de la pierna delantera derecha. Elaboración propia: GeoGebra

como ocurrió anteriormente. En esta captura se resalta en verde las piernas que realizaron la trayectoria de Bézier.

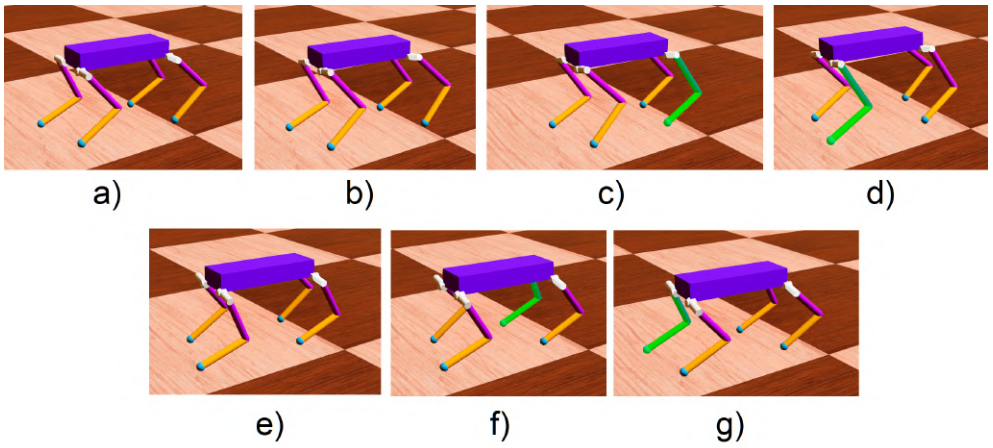


Figura 4.61: Simulación de la secuencia de caminata de gateo. a) Posicionamiento Inicial. b) Movimiento del centro de masa hacia la derecha. c) Balanceo de la pierna trasera izquierda. d) Balanceo de la pierna frontal izquierda. e) Movimiento del centro de masa hacia la izquierda. f) Balanceo de la pierna trasera derecha. g) Balanceo de la pierna frontal derecha Elaboración propia: Webots

4.7.8 Determinación de la posición y orientación del robot

La posición del centro de masa del robot es de especial interés, ya que corresponde a uno de los indicadores principales del desempeño de la secuencia de caminata. A continuación, se presenta una solución a este cálculo, empleando únicamente el posicionamiento de las piernas del robot. Es importante recalcar que el uso de este método en este caso se debe a que la secuencia de caminata desarrollada es relativamente lenta y pausada, por lo cual

se realizan las siguientes suposiciones:

1. Los cambios en la orientación del robot son mínimos.

2. No existe un deslizamiento de las patas con el suelo.

3. Las piernas en la fase de soporte siempre están en contacto con el suelo.

A partir de ello, se puede asumir que cada pata es capaz de indicar la posición actual del centro de masa. Esta posición no es absoluta con respecto al ambiente, sino que es es relativa respecto a un punto de inicio, es decir, se asume que el robot se encuentra en [x = 0, y = 0, z = 0.4m] cuando este se empieza a mover. En Fig. 4.62 se puede observar el robot cuando este se encuentra balanceando una de sus piernas, por lo cual, las otras 3 piernas estarán en la fase de soporte. En base a esto, se podrá obtener 3 mediciones similares del centro de masa en esta fase. Debido a que estas mediciones pueden tener pequeños errores, se decide que la medida final del centro de masa será el promedio de estas medidas de las piernas individuales.

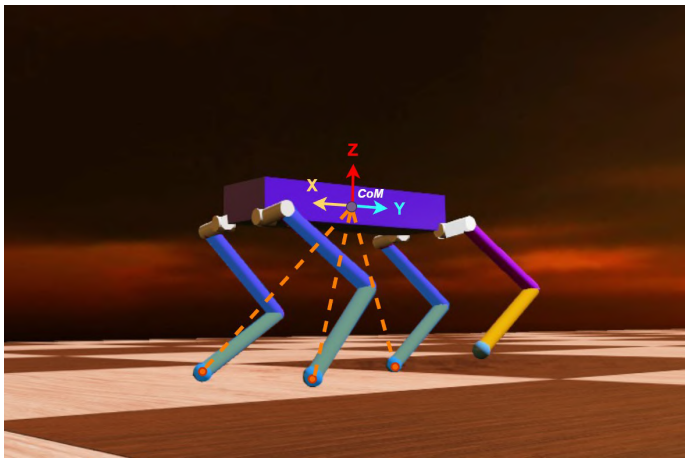


Figura 4.62: Determinación de la posición del centro de masa del robot durante la secuencia de caminado. Elaboración propia: Webots, Diagrams.net

La lógica de la solución de este problema se presenta en el diagrama de Fig. 4.63, en donde se aprecia que los cambios de posición en los 3 ejes (dx, dy y dz), se calculan al observar los cambios apreciados en la posición del efector final en comparación al ciclo anterior. El promedio de dichos cambios se suma a las coordenadas del centro de masa para obtener el resultado deseado.

Para el caso de la orientación, no se recurre a la cinemática, sino que se decide usar el sensor de inercia que posee el robot. Este sensor es capaz de aplicar modos de fusión de manera autónoma, de forma tal que puede medir los ángulos de la orientación del robot.

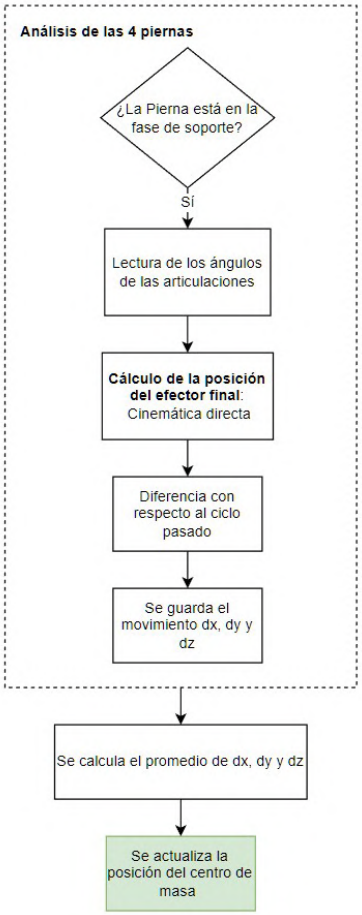


Figura 4.63: Diagrama de la determinación de la posición del centro de masa del robot, Elaboración propia: Diagrams.net

4.7.9 Resultados de la simulación de la secuencia de caminata

Se plantea evaluar el desempeño de la secuencia de caminata en la simulación mediante la observación de la orientación y la posición del centro de masa, sin embargo, también será una referencia para determinar las diferencias con la implementación final en el robot real.

En Fig. 4.64 se muestra la posición del centro de masa del robot, al ejecutar la totalidad de los pasos de la secuencia de caminata. En esta se puede observar que en el eje Z se mantuvo a una altura constante, tal como se esperaba. Por otro lado, el eje Y mostró el comportamiento de mover el centro de masa hacia los lados para incrementar la estabilidad. El eje X representa el avance del robot, el cual tiene un comportamiento incremental y se detiene cuando el Eje Y exhibe un cambio en su posición.

El resultado de la orientación del centro de masa del robot en la simulación se observa en Fig. 4.65. En esta gráfica se aprecia como los cambios percibidos en la orientación son casi nulos en los 3 ejes, llegando a tan solo 1° el eje Pitch, como máximo. Esto comprueba la estabilidad de la secuencia de caminata desarrollada y además, valida la

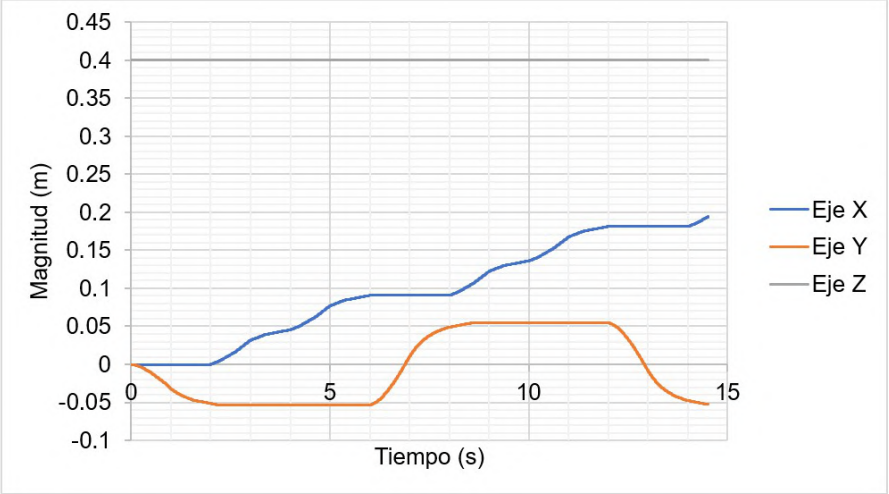


Figura 4.64: Posición del centro de masa del robot durante la simulación de la secuencia de caminado. Elaboración propia: Microsoft Excel

suposición planteada para el cálculo de la posición del centro de masa, la cual consistía en movimientos mínimos en la orientación del robot.

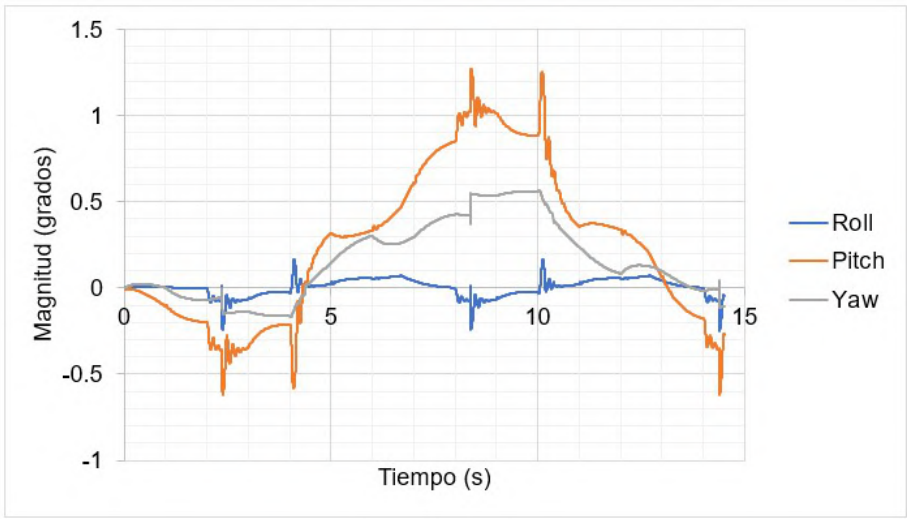


Figura 4.65: Orientación del centro de masa del robot durante la simulación de la secuencia de caminado. Elaboración propia: Microsoft Excel

4.8 Estructura final del software

En este proyecto se han desarrollado la solución a diferentes aspectos del funcionamiento del robot. A la hora de implementarlos en el robot real, todos ellos deberán trabajar al mismo tiempo. Es por ello que surge la necesidad de una arquitectura de control a nivel del software que sea capaz de manejar todos estos procesos en conjunto.

Para ello, se empleará el middleware de ROS2, ya que se espera que el desarrollo de

este robot siga en los próximos años, por lo cual emplear una herramienta que tendrá soporte de una comunidad tan extensa resulta la decisión más óptima. Por otro lado, esta comunidad ha desarrollado controladores para muchos componentes de hardware, por lo cual su uso en términos de modularidad y simplicidad se vuelve de gran interés.

En resumen, las tareas que se deben organizar en la arquitectura de control corresponden a:

1. Interfaz de hardware del control para las pruebas
2. Interfaz de hardware de la comunicación de los motores
3. Interfaz de hardware del sensor de inercia (IMU)
4. Interfaz de hardware para interconectar la computadora central con la de percepción
5. Procesamiento de la detección de contacto
6. Procesamiento de la determinación de la posición del centro de masa
7. Procesamiento de las trayectorias generadas por la secuencia de caminata
8. Procesamiento del algoritmo de control de las piernas.

Como se ha mencionado, las arquitecturas híbridas corresponden a las más robustas, siendo capaces de implementar tareas reactivas de baja nivel y tareas de planeamiento de alto nivel. Debido a la complejidad del funcionamiento del robot cuadrúpedo, esta resulta la mejor decisión para su desarrollo.

Sin embargo, este proyecto está enfocado principalmente a la locomoción, por lo cual resulta difícil especificar y validar tareas de planeamiento que tomen en cuenta la percepción. Debido a ello, este tipo de tareas se plantearán únicamente en el esquema general, pero no serán de interés en la implementación y validación. De esta manera, se tendrá una capa reactiva, correspondiente a la locomoción, la cual controla los actuadores en base a los sensores y que además, recibirá comandos de capas de mayor nivel. En las capas superiores, se encontrará la generación de trayectorias y la lógica de la secuencia en un nivel medio y la percepción del ambiente en un nivel alto. El nivel de las capas está ligado a la frecuencia a las que estas funcionarán, es decir, la locomoción deberá tener una frecuencia relativamente alta, mientras que la percepción debe ser mucho más baja debido a la carga computacional que representa.

En Fig. 4.66 se puede observar la estructura de la arquitectura de control híbrida que implementa la totalidad de los módulos desarrollados. En esta se demarcan con rojo los que no forman parte de este proyecto y con gris los componentes de hardware. Adicionalmente, se especifica la frecuencia aproximada a la que trabajan y los mensajes que envían y reciben.

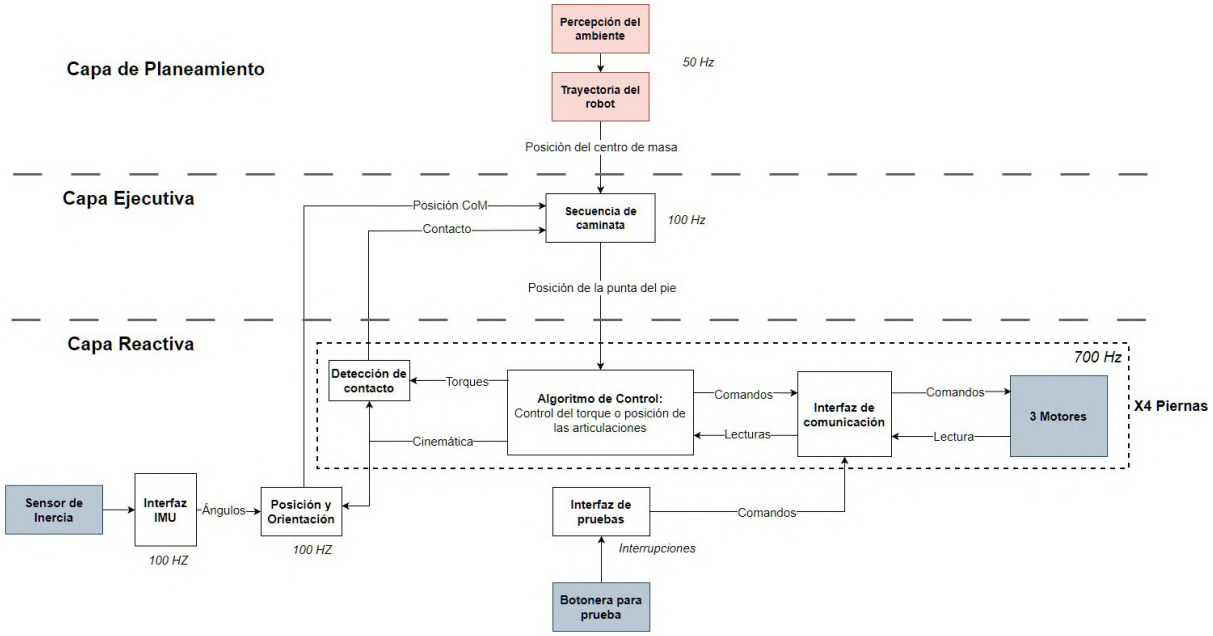


Figura 4.66: Estructura de la arquitectura de control híbrida. Elaboración Propia: Diagrams.net

Capítulo 5

Resultados y Análisis

En este capítulo se presenta la validación final de la arquitectura de control. En el capítulo anterior se realizaron validaciones previas de los módulos individuales, sin embargo, se debe evaluar el rendimiento de todos en conjunto, así como emplear la estadística para respaldar los resultados y las especificaciones del robot mencionadas. Primeramente, se evaluará la frecuencia a la que se podrá controlar el robot, para después evaluar el rendimiento de la secuencia de caminata en el robot real.

5.1 Validación de la frecuencia de control

En esta sección se presentan los resultados iniciales del desempeño de la frecuencia de comunicación, para el cual se selecciona el método para gestionar los mensajes y el lenguaje de programación. Posteriormente, se realiza un análisis estadístico de la solución.

5.1.1 Pruebas preliminares

Para validar el diseño de la red de comunicación, se plantea enviar 1000 mensajes en cada uno de los 4 canales y esperar la respuesta de los actuadores. Para ello, se medirá el tiempo de cada ciclo de muestreo y se calculará la frecuencia individual para cada uno de ellos.

En Fig. 5.1 se muestran los resultados de frecuencia de muestreo de la prueba planteada. En ella se pueden ver 4 gráficos de caja, los cuales poseen una media de alrededor de 600 Hz. Por otro lado, se puede notar la existencia de valores atípicos en los 4 casos, llegando hasta frecuencias cercanas a los 100Hz. Estas variaciones se pueden deber por diversas razones, dentro de las cuales se pueden considerar la latencia entre el procesador y los módulos CAN, la falta de una sincronización en la comunicación de los motores e incluso las variaciones intrínsecas que se dan al ejecutar un código en un sistema operativo. Por otro lado, es importante recalcar que las muestras correspondían a 1000 datos y dichos

valores típicos no representan ni siquiera el 5% de los mismos. A pesar de ser valores atípicos, estos se encuentran por encimar de 100 Hz, por lo cual, no representan un bloqueo severo en la ejecución de la interfaz. Con el fin de reducir estas variaciones atípicas de la frecuencia, se propone evaluar el rendimiento del software de programación empleando el lenguaje C++ en lugar de Python.

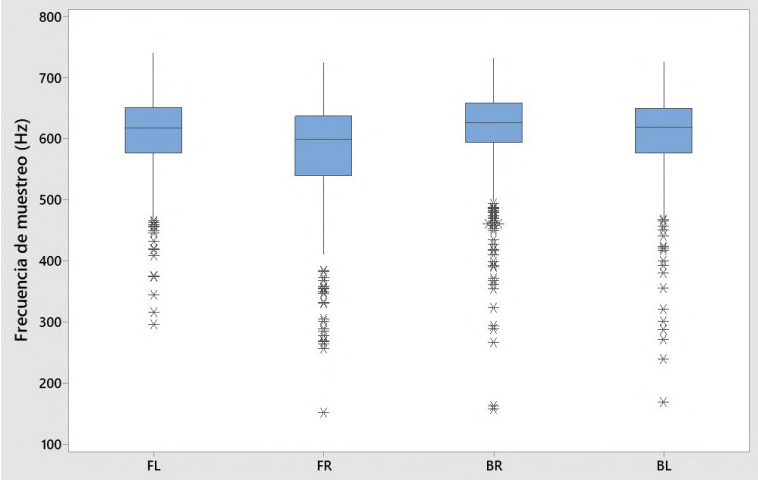


Figura 5.1: Gráfica de cajas de la frecuencia de muestreo para la comunicación de las 4 piernas del robot. Elaboración propia: Minitab

El lenguaje de programación de C++ resulta rápido y eficiente, por lo cual es posible que se logre disminuir las latencias entre la comunicación del motor y el procesamiento de los mensajes. A partir de ello, se realiza la misma prueba anterior pero en este lenguaje, tal como se presenta en Fig. 5.2, en el cual se nota que la media de la frecuencia de muestreo aumentó considerablemente, además que los valores atípicos no llegan a valores tan bajos y se presentan una menor cantidad de ellos. En base a esto, se determina que el lenguaje más apropiado para la interfaz de comunicación corresponde a C++.

Al investigar la documentación del motor, se encuentra que existe otra manera de establecer la comunicación con múltiples motores. Esta consiste de un modo transmisión de mensajes en el cual múltiples motores pueden procesar un comando a partir de un único mensaje de la computadora. En Fig. 5.3 se observa la lógica de esta comunicación, y la manera en que los motores responden en orden de acuerdo con la prioridad propia de su identificador. De esta manera, se consigue una comunicación mucho más ágil que la planteada anteriormente. Sin embargo, en este modo solo se puede transmitir comandos de torque, por lo cual, procesos que requieran un control de posición o cambiar parámetros del motor no serán capaces de efectuarse.

Al evaluar el desempeño de este método al enviar y recibir mensajes durante 1000 ciclos, empleando el lenguaje de programación C++, se obtiene el diagrama de cajas en Fig. 5.4. En este se puede apreciar como las medidas se concentran significativamente alrededor de la media, indicando una gran estabilidad en el proceso. Si bien se presentan valores atípicos, estos resultan muy pocos e incluso suceden para valores de frecuencia más altos que la media.

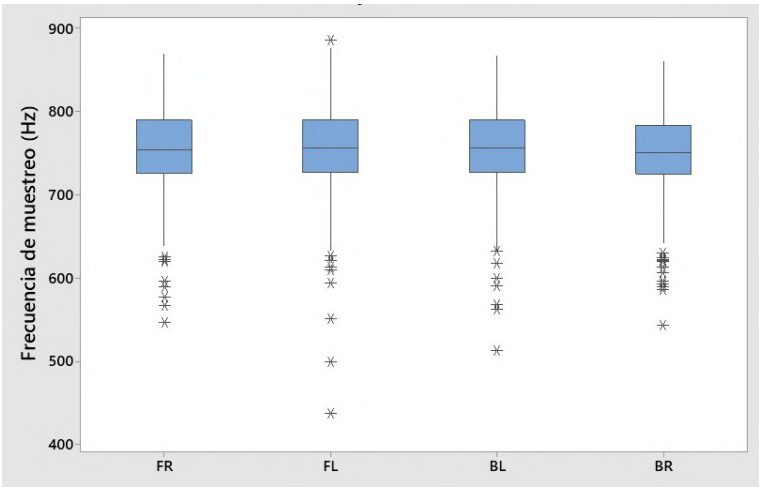


Figura 5.2: Gráfica de cajas de la frecuencia de muestreo para la comunicación de las 4 piernas del robot empleando el lenguaje de programación C++. Elaboración propia: Minitab

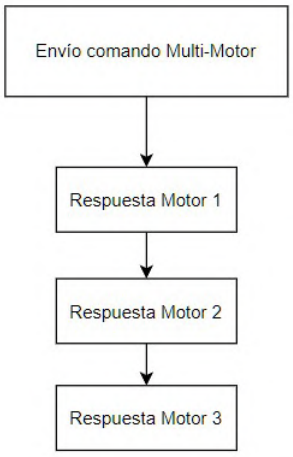


Figura 5.3: Diagrama de la comunicación Multi-Motor. Elaboración propia: Diagrams.net

En la tabla 5.1 se puede observar el resumen de la media de frecuencia de muestreo y la desviación estándar de las 3 soluciones propuestas en esta validación para mejorar el candidato. El lenguaje de Python a pesar de permitir una gran flexibilidad para modificar el código en un tiempo corto, posee un desempeño relativamente bajo en comparación a los dos candidatos. El candidato con lenguaje de programación C++ con la lógica desarrollada anteriormente y el candidato de la comunicación multi-motor poseen medias de frecuencia relativamente similares, sin embargo, este último presenta una estabilidad superior que se demuestra con el menor valor desviación estándar.

A pesar de ello, la poca flexibilidad en cuanto a los tipos de comandos que se le pueden enviar al motor resulta una desventaja muy significativa. Esto debido a que requeriría que el robot estuviera en un control de torque de manera permanente, sin embargo, estos controles son los que se planean implementar en el robot en un futuro, por lo cual, en este momento no se posee la capacidad para aprovechar este modo. Es por ello que se decide

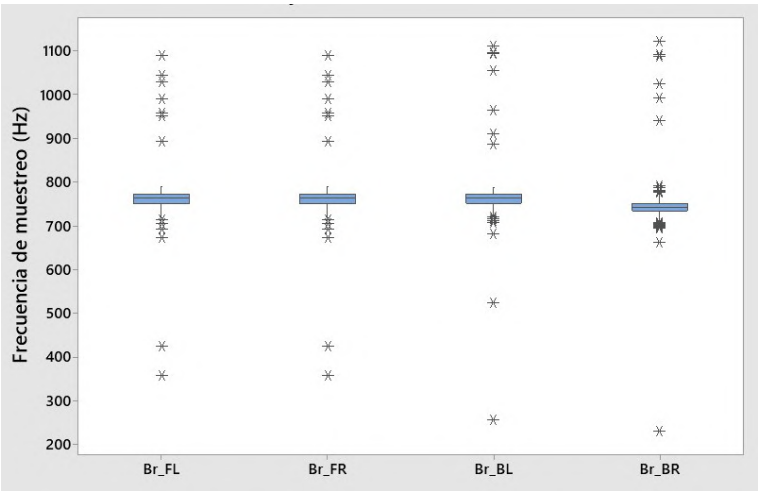


Figura 5.4: Gráfica de cajas de la frecuencia de muestreo para la comunicación de las 4 piernas del robot empleando el lenguaje de programación C++ y el comando de transmisión multi-motor. Elaboración propia: Minitab

Solución	Media (Hz)	Desv. estándar (Hz)
Python	585.182	71.156
C++	755.061	47.133
C++ y Transmisión Multi-Motor	761.655	30.267

Tabla 5.1: Comparativa del rendimiento de la comunicación de los motores

emplear el candidato que emplea C++ y la lógica desarrollada, tomando como referencia la capacidad de frecuencia de muestreo y la funcionalidad.

Importancia del poder computacional en la comunicación

Actualmente se está trabajando con una Raspberry Pi 4B como computadora central, sin embargo, el laboratorio de investigación tiene a disponibilidad una Raspberry Pi 5. Por lo tanto, se realizan pruebas en este dispositivo para determinar si existe una mejora en el desempeño y si merece la pena implementarse en este proyecto. Dentro de sus ventajas, se incluye una capacidad de procesamiento mayor al doble que su antecesora, esto a un precio sumamente similar a su competidor [2].

En Fig. 5.5 se muestra el diagrama de cajas de las pruebas realizadas. En este se observa que los valores atípicos se redujeron significativamente. Su distribución es similar a la de la RPI4, esto sin llegar a concentrar sus valores de frecuencia de la misma manera que la transmisión multi-motor. Al calcular la media de las muestras se obtiene que la frecuencia de control alcanza los 775.552 Hz, con una desviación estándar de 39.672 Hz. De esta manera, se determina que la comunicación en la RPI5 es ligeramente superior a la de la RPI4, incluso reduciendo los valores anormales presentes de la ejecución. Esta mejora de desempeño es de apenas un 3% aproximadamente, por lo cual, la decisión de cambiar la computadora central no se puede respaldar de manera significativa por este criterio.

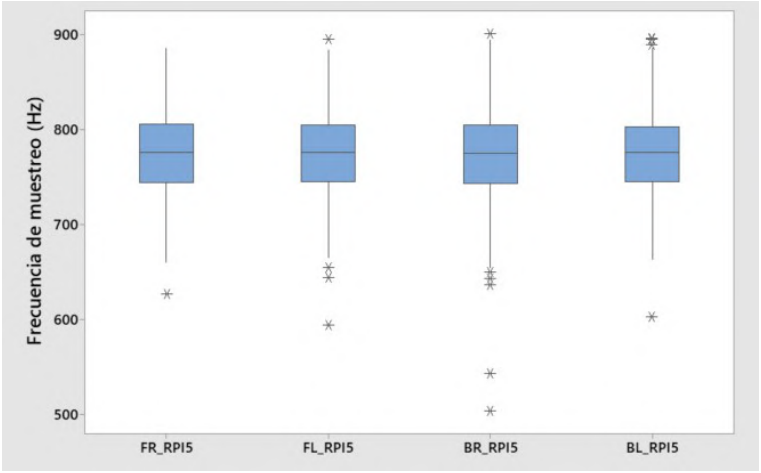


Figura 5.5: Gráfica de cajas de la frecuencia de muestreo para la comunicación de las 4 piernas del robot empleando el lenguaje de programación C++ en una Raspberry Pi 5. Elaboración propia: Minitab

5.1.2 Validación de la interfaz de comunicación en la arquitectura de control

Las pruebas preliminares se realizaron cuando la computadora central no tenía ninguna carga de procesamiento adicional. Si bien se puede evaluar el desempeño con la secuencia de caminata planteada, resulta conveniente emplear uno de los controles dinámicos que se desea implementar en un futuro y que resultan mucho más demandantes que los presentados acá.

Esta prueba consistirá en enviar y recibir un gran número de mensajes a las 4 piernas del robot, mientras se procesa el algoritmo de control a una frecuencia de 750 Hz, valor que se propone como el máximo en base a la media de frecuencia de muestreo sin carga computacional. Esta implementación del control no es funcional en el sentido de que los actuadores no ejecutan ninguna acción física. En su lugar, se llevan a cabo cálculos complejos y se actualizan las lecturas de los motores. De esta manera, se simula el rendimiento del hardware en una futura implementación de dicho control. Estos procesos se realizan en paralelo, debido a que se plantea una comunicación entre los mismos empleando espacios compartidos de la memoria.

El control a emplear corresponde al VMC, el cual involucra el cálculo de múltiples comportamientos de resorte-amortiguador en el robot y además el cálculo de fuerzas que debe ejercer cada una de las patas. Esto se realiza en base a una simulación en el lenguaje de programación C++, por lo cual resulta relativamente simple implementarlo en el código actual. Si bien el control MPC resulta más demandante en cada ciclo, su frecuencia de muestreo es mucho menor que la del VMC y además, sus simulaciones actualmente no se pueden implementar directamente en el robot.

Como se mencionó en las especificaciones, se espera que el robot pueda tener una frecuencia de muestreo de más de 500 Hz. Por otro lado, la validación preliminar del sistema de

comunicación mostró una media de muestreo de alrededor de 750 Hz, sin embargo, para compensar posibles retrasos por el gran número de cálculos, se espera que sea mayor a 700 Hz. No se establece un límite superior debido a que valores muy altos de frecuencia no afectan negativamente al sistema. En base a dichas características, se evaluará si es posible obtener una tasa de muestreo en donde el 95% de los datos sea mayor a 500 Hz y que se llegue a obtener una media superior a 700 Hz.

Para determinar el tamaño de la muestra se emplea la ecuación 5.1, la cual se basa en una distribución normal [39, p. 298].

$$n = \frac{Z^2_{\alpha/2} \cdot \hat{p} \cdot \hat{q}}{e^2} \tag{5.1}$$

En ella, $Z^2_{\alpha/2}$ representa el estadístico critico de la distribución normal, para el cual se emplea 1.645, valor correspondiente al 95% de confianza unilateral. Por otro lado, $\hat{p}$ se emplea como el estimado de p, el cual se puede calcular previamente usando una muestra de $n \geq 30$, en dicho caso se emplearon 30 datos de frecuencia, en donde el 87% superaba el valor de 700 Hz. Asimismo, $\hat{p}$ consiste en la proporción complementaria de la población, la cual sería 0.13, considerando el valor de $\hat{p}$ anterior. Por último, e corresponde al error de la estimación, el cual se asume en 5%. A partir de ello se calcula el número de muestras necesarias, tal como se muestra en la ecuación 5.2, resultando en al menos 123 datos de frecuencia necesarios para el análisis estadístico.

$$n = \frac{1.645^2 \cdot (0.87) \cdot (0.13)}{(0.05)^2} = 122.42 \tag{5.2}$$

Posteriormente se comprueba la normalidad de los datos, para lo cual se realiza la prueba de Anderson-Darling. Debido a que en este experimento se usa una confiabilidad del 5%, el valor de p debe ser mayor a 0.05 para que la aproximación de normalidad sea aceptable. El resultado de esto se puede ver en Fig. 5.6, en donde se observa que se obtuvo un valor de p de 0.574, por lo cual, se concluye que la población sí se puede aproximar con una distribución normal.

Asimismo, se determina el intervalo de tolerancia unilateral con un 95% de confianza, de manera que se pueda visualizar en dónde se encuentra la mayoría de los datos. En Fig. 5.7 se puede observar que el límite inferior corresponde a 657.33 Hz, lo cual corresponde alrededor de un 12% de variación con respecto al valor de la media. De esta manera, se puede inducir que la mayoría de los datos están ubicados correctamente, tomando en cuenta el valor mínimo de 500 Hz y al observar que la forma del histograma presenta una simetría.

Debido a que se posee un límite de especificación y una media esperada, se decide emplear un análisis de capacidad. A partir del supuesto de aproximación normal obtenido anteriormente, se realiza el análisis de capacidad normal mostrado en Fig. 5.8.

En este análisis se puede observar como el histograma muestra que la mayor parte de los

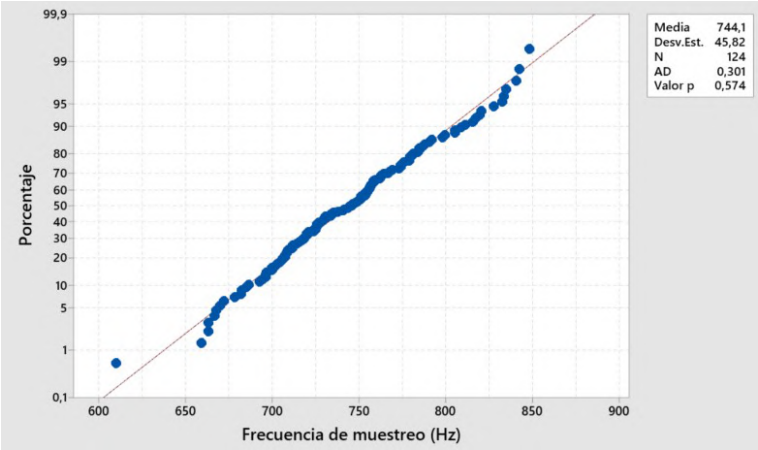


Figura 5.6: Prueba de normalidad de las muestras de frecuencia de control. Elaboración Propia: Minitab

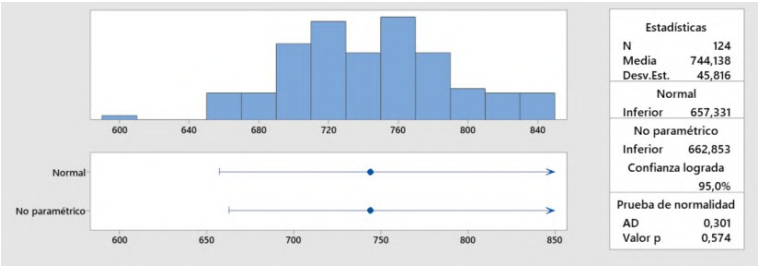


Figura 5.7: Intervalo de tolerancia de la frecuencia de muestreo. Elaboración Propia: Minitab

datos logran superar el límite inferior establecido de 500 Hz. Por otro lado, se observa como la campana está centrada en un valor superior al objetivo de 700 Hz, lo cual se refleja en la media de 744.14 Hz.

Asimismo, se puede observar que el parámetro Cpk, o el índice de capacidad real, muestra un valor de 1.72, lo que indica que a corto plazo el proceso tiene la capacidad de producir productos dentro de las especificaciones. Por otro lado, el parámetro Ppk, o el índice de capacidad potencial, tiene un valor de 1.78, confirmando que se podrán cumplir las especificaciones teniendo en cuenta la variabilidad del proceso y los límites de especificación. Además, el índice de capacidad potencial de rendimiento (Cpm) resulta ser de 1.05, lo cual indica que se puede esperar un rendimiento ligeramente mejor de lo anticipado. A partir de ello, se comprueba que los resultados de frecuencia de muestreo cumplen con las necesidades.

Comparación del rendimiento de las computadoras centrales

En el desarrollo de la solución, se encontró que cambiar la computadora central no resultaba en mejoras notorias en la frecuencia de comunicación, sin embargo, ahora conviene volver a evaluar el resultado de otras opciones ya que se están realizando las pruebas con un algoritmo de control lo suficientemente complejo. Para ello, se empleará la Raspberry

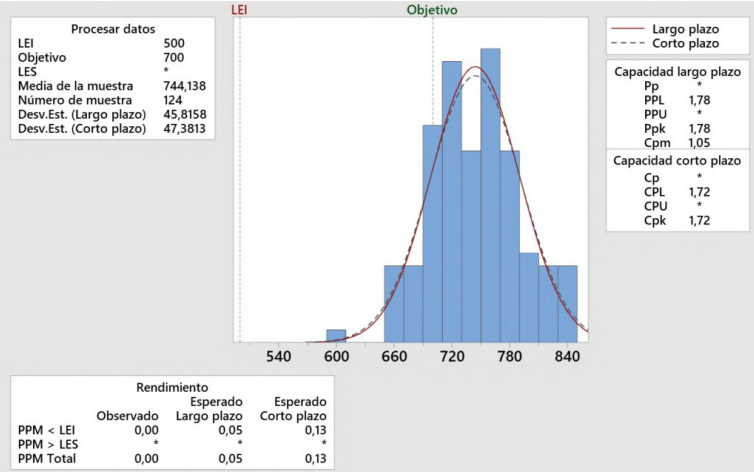


Figura 5.8: Análisis de capacidad de la frecuencia de control. Elaboración Propia: Minitab

Pi 5 y una computadora portátil con un procesador Intel Core i7 8750H y Ubuntu como sistema operativo. Al ejecutar el mismo código en dichos procesadores, se obtienen los resultados de la tabla 5.2. En la misma se aprecia la superioridad de la Raspberry Pi 5 con respecto a la Laptop, lo cual se puede deber a una menor latencia en su hardware. En este caso, la mejora respecto la RPI 4 y la RPI 5, es de tan solo un 6%.

A partir de ello se puede concluir que si bien dicha mejora no es considerablemente alta y tampoco es necesaria porque la RPI 4 ya cumple con los requisitos, esta computadora central podrá ser de utilidad cuando se desee implementar algoritmos más complejos y el rendimiento de la RPI 4 se vea afectado notablemente. Esto debido a la compatibilidad del código desarrollado entre ambas plataformas, ya que se puede implementar directamente.

Computadora	Media de frecuencia (Hz)	Desv. Estándar (Hz)
Raspberrry Pi 4	744.2	45.82
Raspberrry Pi 5	786.7	38.56
Laptop	774.3	51.61

Tabla 5.2: Comparación del rendimiento de las computadoras centrales.

5.2 Evaluación de la comunicación con el sensor de inercia

La estimación de estados del robot requiere de la información proporcionada por el sensor de inercia, la cual incluye aceleraciones lineales, velocidades angulares y ángulos. En base a esto, se induce que la frecuencia máxima a la que trabajará dicho módulo de estimación es dependiente de la frecuencia de muestreo del sensor. Por otro lado, los bloques de mayor nivel en otros algoritmos, como lo es el MPC, requieren de la estimación de estados para

calcular la fuerza requerida en cada una de las piernas, es decir, el sensor también será una limitante en este módulo. Debido a ello, se debe analizar su capacidad en la arquitectura empleada.

Si bien en la hoja de datos se establece una tasa de muestreo de 100 Hz [8], el mismo componente posee un microprocesador, el cual se encarga de calibrar y gestionar la información de los sensores. Es por ello que, se comprobará la velocidad de comunicación entre este y la computadora central, de manera que se pueda validar que el algoritmo de control tendrá disponible el muestreo a 100 Hz.

Para realizar dicho experimento, se toman medidas continuas durante 5 min, lo cual generará un gran número de datos, sin embargo, también se desea ver si existe una total recepción de los paquetes. Para analizar dichos datos, primeramente se comprueba si poseen una distribución normal, pero tal como se ve en Fig. 5.9, el valor de p del ajuste de bondad es mucho menor a 0.05, lo cual indica un incumplimiento. Al emplear la herramienta de Minitab para la identificación de distribuciones, se encuentra que tampoco se cumple con las demás presentes en el software. En base a esto, se decide emplear métodos no paramétricos.

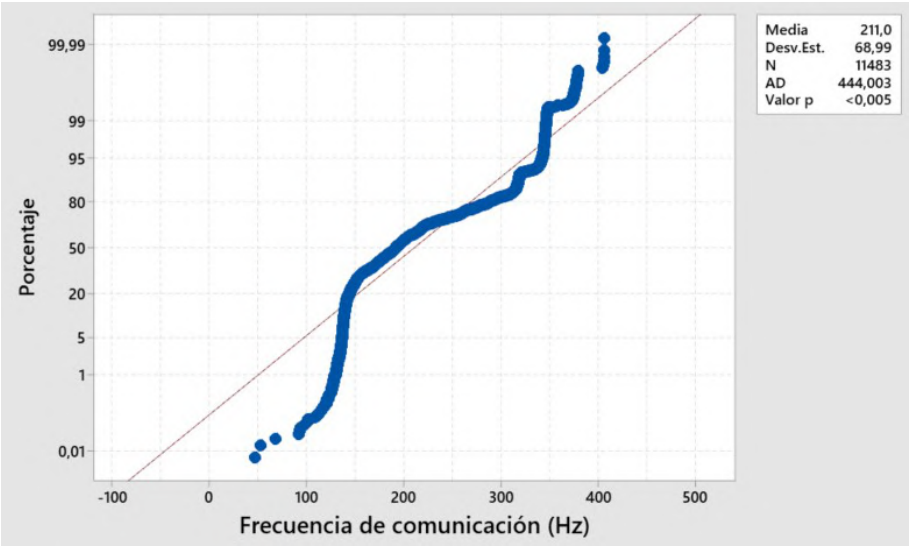


Figura 5.9: Prueba de normalidad para los datos del IMU. Elaboración Propia: Minitab

Posteriormente, se evalúa en dónde están distribuidos los datos, esto mediante el intervalo de tolerancia no paramétrico de Minitab. De esta manera, al emplear un porcentaje de confianza de 95% bilateral, se obtiene Fig. 5.10, en la cual se observa su histograma en la parte superior y el gráfico del intervalo en la parte inferior. En ella se observa que se obtuvo una media de 211.0 Hz, mientras que el rango de confianza corresponde a [133.9 Hz, 345.3 Hz]. A pasar de que los resultados obtenidos sean mayores a 100 Hz, estos solo representan la velocidad de comunicación del procesador interno del sensor y la computadora central, es decir, la frecuencia de muestreo de los sensores seguirá siendo 100 Hz, por lo cual se toma dicho valor como el verdadero.

Para confirmar finalmente el cumplimiento del requisito, se realiza una prueba de hipótesis

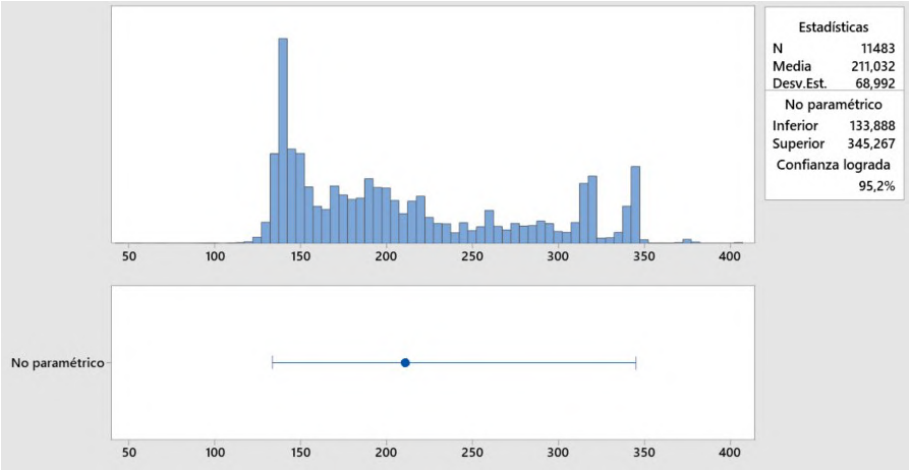


Figura 5.10: Intervalos de tolerancia de la frecuencia de comunicación del IMU. Elaboración Propia: Minitab

mediante la prueba de signo no paramétrica. Las hipótesis se definen de acuerdo con los rangos mencionados anteriormente, tal como se presenta en la ecuación 5.3.

$$\text{Hipótesis: } \begin{cases} H_0 : \mu_{\text{frec.}} \leq 100 \text{ Hz} \\ H_1 : \mu_{\text{frec.}} > 100 \text{ Hz} \end{cases} \tag{5.3}$$

Mediante el software de Minitab, se obtiene que el valor de p corresponde a 0.000, el cual al ser menor a 0.05, indica que la evidencia estadística es suficiente para descartar la hipótesis nula en favor de la hipótesis alternativa. A partir de ello, se concluye que el sistema sí es capaz de trabajar con los datos del sensor de inercia a 100 Hz.

5.3 Validación de la interfaz de control

Para validar el funcionamiento de la interfaz de control, se evalúo el funcionamiento de la actuación de los botones, debido a que consisten el elemento principal de este apartado. Para ello, se busca probar el correcto funcionamiento de las interrupciones implementadas en la computadora central para cada uno de los botones. En base a ello, se desarrolla una prueba la cual consiste en activar cada uno de los botones secuencialmente, de manera que se inicia con el botón de “Start” y se termina con el “Extra”.

En Fig. 5.11 se muestra las banderas de activación de los 5 botones, la cuales van desde 0 hasta 1. Una vez estas son activadas, se mantienen en dicho valor, debido a que la computadora central deberá ejecutar una acción antes volver a detectar la activación de dicho botón. En esta prueba se puede ver como los 5 botones fueron accionados y procesados correctamente por la función de interrupciones de la computadora central. En base a esto, se determina que tanto los componentes físicos, el circuito desarrollado y la programación, resultan en una interfaz de control funcional, la cual está lista para ser

implementada en la arquitectura.

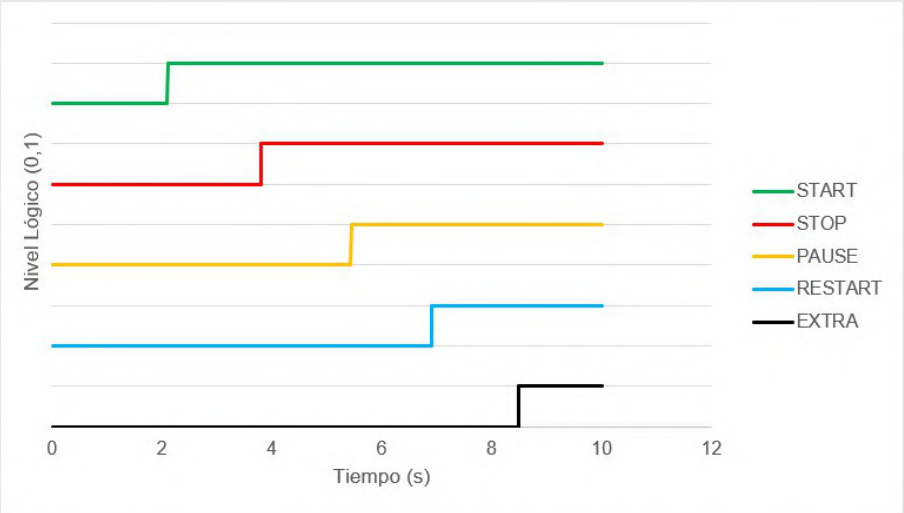


Figura 5.11: Prueba de detección de los botones de la interfaz de prueba. Elaboración propia: Microsoft Excel

5.4 Validación de la detección del contacto

Con el fin de evaluar la solución planteada a la detección de contacto, se planteó una prueba en la que una de las piernas posee una trayectoria predefinida, sin embargo, se sitúa un obstáculo que interfiere con la misma. Por otro lado, se busca detener el robot cuando este detecte el obstáculo, ya que si se continua la trayectoria, el mismo se apoyará sobre el obstáculo e intentará levantarse más de lo deseado, desestabilizando el cuerpo del robot.

En Fig. 5.12.a se puede observar la configuración empleada para la prueba. La pierna inicia en el aire, debido a que se buscaba facilitar una diferencia de alturas entre el inicio y el contacto con el obstáculo. Posteriormente esta se levanta, tal como se ven en Fig. 5.12.b y en Fig. 5.12.c se contacta la caja. A partir de ello, la detección muestra ser lo suficientemente rápida como para detener la pierna justo a tiempo y evitar que se levante el cuerpo, tal como se ve en la posición final enFig. 5.12.d.

Por otro lado, en Fig. 5.13 se observa que el comportamiento de los torques cambia ligeramente en comparación a los estudios previos. En base a ello, se puede notar como el torque de la rodilla no alcanza valores sumamente altos, como los que se mostraba en el impacto de Fig. 4.41. Además, se nota como la detección ocurre justo antes de llegar al valor máximo de dicho torque, lo cual comprueba la agilidad de la solución planteada.

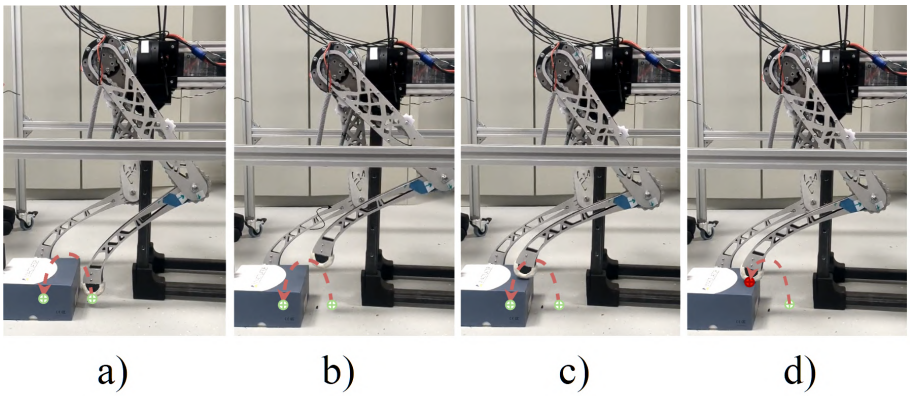


Figura 5.12: Detección de un obstáculo en la trayectoria. a) Punto inicial. b) Levantamiento de la pierna. c) Contacto con el objeto. d) Detección y freno de la trayectoria. Elaboración Propia: Diagrams.net

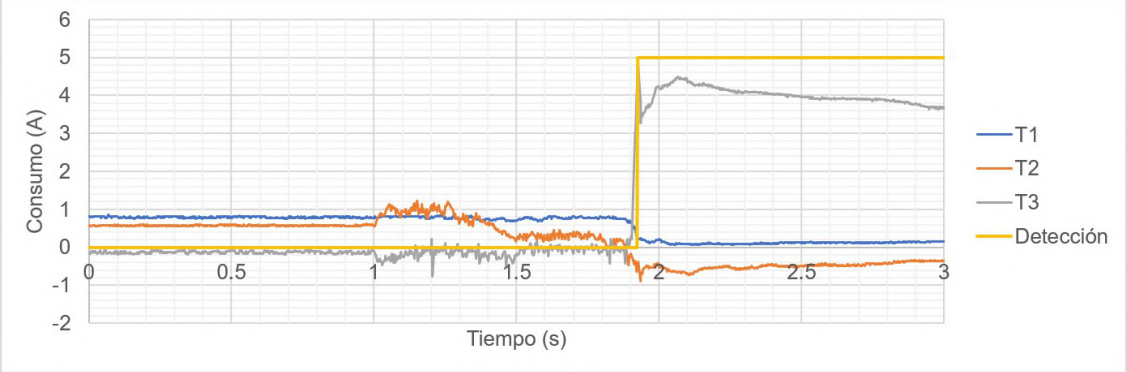


Figura 5.13: Comportamiento del Torque en las articulaciones durante el balanceo con un obstáculo. Elaboración Propia: Microsoft Excel

5.5 Evaluación del desempeño de la secuencia de caminata

Para evaluar la secuencia de caminata empleada, se plantea ejecutar un ciclo completo de la misma, en la cual las 4 de las piernas del robot realizaran un balanceo. De esta manera, se podrá analizar el rendimiento completo de la misma. En Fig. 5.14 se presenta el movimiento del robot físico, mostrando los movimientos del centro de masa hacia los lados y el balanceo de las 4 patas. Es importante mencionar que el robot posee cuerdas sujetas a una estructura metálica que se mueve a medida que el robot lo hace, esto con el fin de prever cualquier caída que dañe gravemente al mismo. Sin embargo, los mismos no se encuentran tensionados cuando se realiza la secuencia de caminata, por lo cual no se afectará el rendimiento.

La posición del centro de masa durante el recorrido se puede ver en Fig. 5.15, en la misma también se presenta el resultado de la simulación obtenido anteriormente. Se observa que los resultados muestran un comportamiento muy similar, sin embargo, los

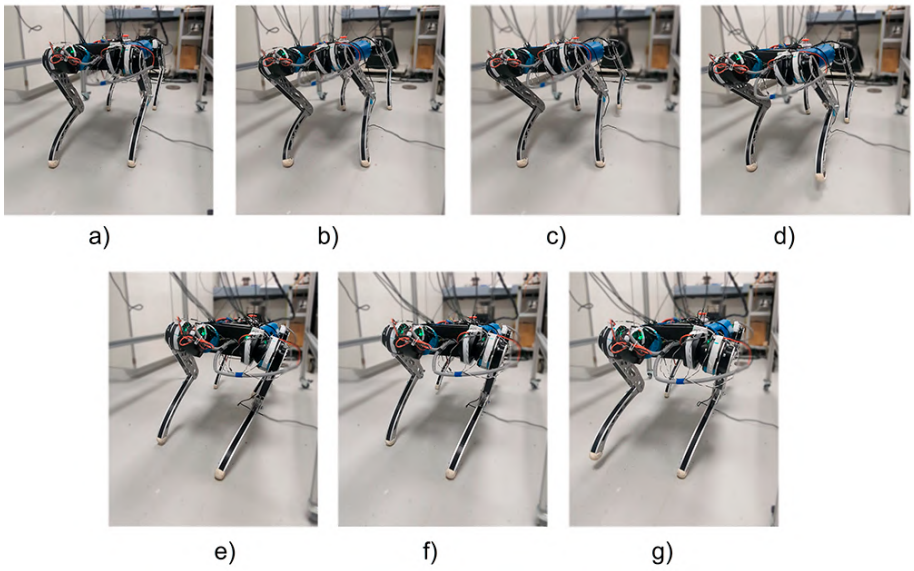


Figura 5.14: Implementación de la secuencia de caminado en el robot real. a) Posicionamiento Inicial. b) Movimiento del centro de masa hacia la derecha. c) Balanceo de la pierna trasera izquierda. d) Balanceo de la pierna frontal izquierda. e) Movimiento del centro de masa hacia la izquierda. f) Balanceo de la pierna trasera derecha. g) Balanceo de la pierna frontal derecha. Elaboración Propia: Microsoft Excel

datos experimentales son relativamente lineales en comparación a los teóricos, los cuales por ejemplo, muestran curvas cuando se mueve el centro de masa en el Eje Y. Esto se debe a la diferencia de los controles PID de la simulación y de la realidad.

Por otro lado, los resultados de la orientación obtenidos a través del sensor de inercia son mostrados en Fig. 5.16. En este mismo se observan cambios mucho más bruscos en la realidad, lo cual se puede atribuir a un control más rígido de los motores, que desequilibran momentáneamente el cuerpo al no amortiguar adecuadamente los impactos. Dicho problema se tratará más adelante.

Como se mencionó anteriormente, se plantea evaluar la totalidad de la secuencia, por lo cual, se usó todas las muestras obtenidas en la misma. Para reducir el número de datos a analizar, se reduce la frecuencia de toma de datos a 50 Hz, por lo cual, de manera que se obtienen alrededor de 800 datos en total. Al realizar la prueba de normalidad de Anderson-Darling para cada una de las variables, se obtiene los resultados de la tabla B.1, en la misma se nota que los valores de p corresponden a "< 0.005", lo cual indica que ninguna de las poblaciones se ajusta a una distribución normal. Esto mismo ocurre al analizar la bondad de ajuste con las demás distribuciones del software de Minitab.

También vale la pena observar gráficamente la distribución de los valores obtenidos del error. En Fig. 5.17 se presenta el intervalo de tolerancia del eje X, el cual tiene un límite superior de 0.019 m, además de que su distribución no parece tan simétrica, sino que se concentra en valores bajos. En el eje Y, mostrado en Fig. 5.18, se presenta que límite del intervalo corresponde a 0.023 m, asimismo, su distribución también muestra

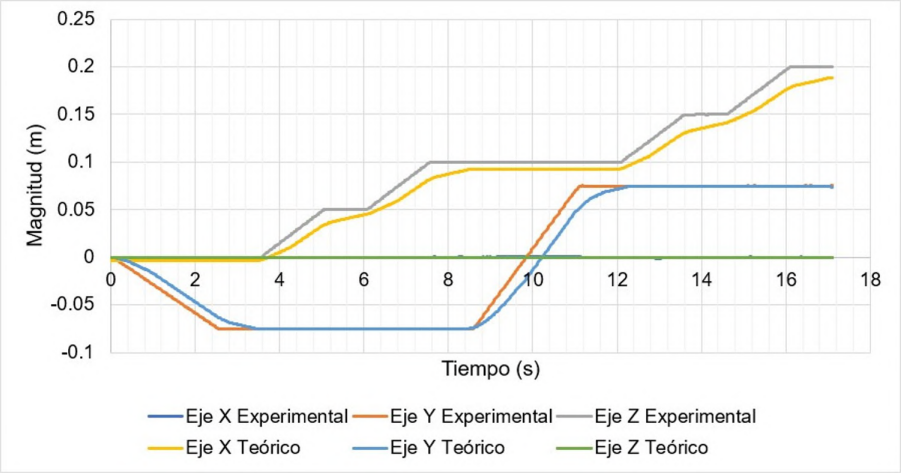


Figura 5.15: Comparación de los resultados experimentales y teóricos de la posición del centro de masa del robot. Elaboración Propia: Microsoft Excel

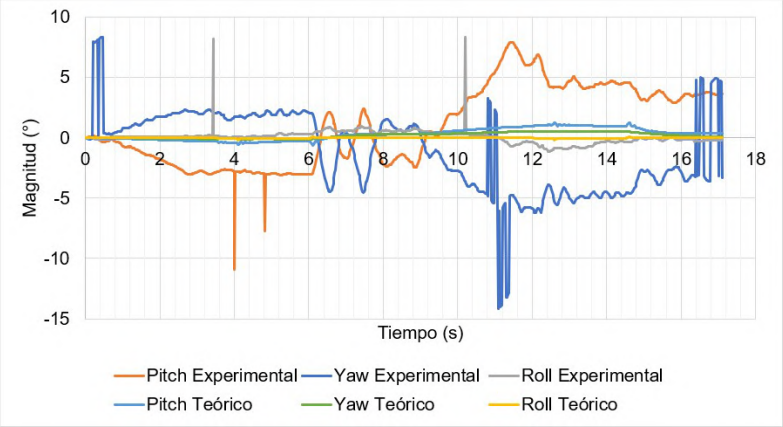


Figura 5.16: Comparación de los resultados experimentales y teóricos de la orientación del centro de masa del robot. Elaboración Propia: Microsoft Excel

una concentración en los valores más bajos. En el caso del Eje Z, se obtuvo un límite de error de 0.001 m y además su distribución es asimétrica, tal como se observa en Fig. 5.19.

Tabla 5.3: Resultados de las pruebas de normalidad de Anderson-Darling para la estimación de estados del centro de masa del robot

Variable de Prueba	**Media de la muestra**	**Desviación estándar**	**Valor p**
Error en el Eje X (m)	0.01028	0.005273	<0.005
Error en el Eje Y (m)	0.005134	0.007364	<0.005
Error en el Eje Z (m)	0.0001602	0.0001903	<0.005
Error en Roll (°)	0.3554	0.4577	<0.005
Error en Pitch (°)	2.497	1.427	<0.005
Error en Yaw (°)	3.079	2.127	<0.005

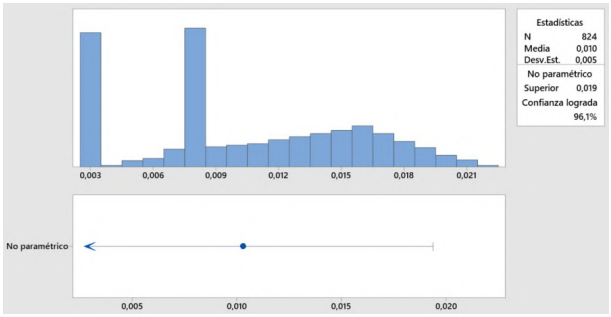


Figura 5.17: Intervalos de tolerancia no paramétricos del error en el Eje X. Elaboración Propia: Minitab

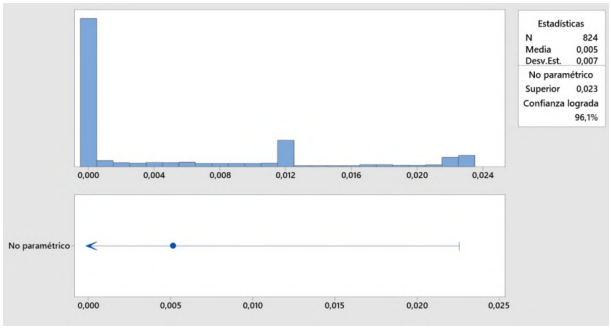


Figura 5.18: Intervalos de tolerancia no paramétricos del error en el Eje Y. Elaboración Propia: Minitab

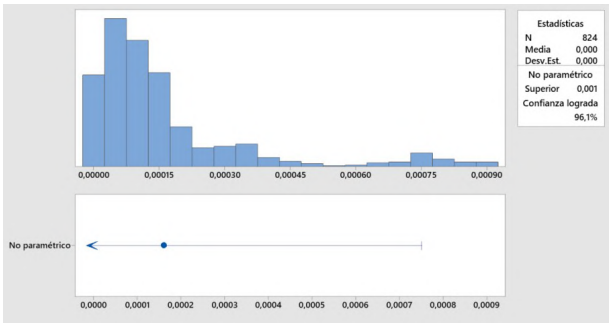


Figura 5.19: Intervalos de tolerancia no paramétricos del error en el Eje Z. Elaboración Propia: Minitab

Para el caso de las orientaciones, se observan límites superiores en los intervalos de tolerancia de 5.664°, 0.849° y 6.449° para los ejes Pitch, Roll y Yaw, respectivamente. Además, la distribución de la muestra del eje Pitch se ve relativamente simétrica, tal como se ve en Fig. 5.20. Por otro lado, en Fig. 5.21 se ve una asimetría en el eje Roll y finalmente, en Fig. 5.22 se observa una distribución simétrica.

En base a esta inconsistencia en la simetría de las distribuciones y que ninguna se ajusta a las distribuciones conocidas del software, se emplea la prueba signo, debido a que los histogramas de los errores mostraron ser no necesariamente simétricos [p. 656][39].

Para realizar el análisis estadístico mencionado, primero se deben definir las hipótesis de

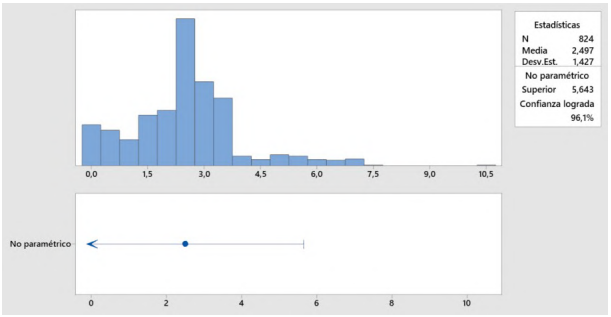


Figura 5.20: Intervalos de tolerancia no paramétricos del error en el Eje Pitch. Elaboración Propia: Minitab

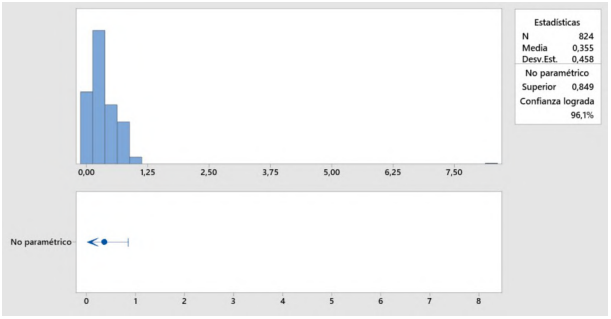


Figura 5.21: Intervalos de tolerancia no paramétricos del error en el Eje Roll. Elaboración Propia: Minitab

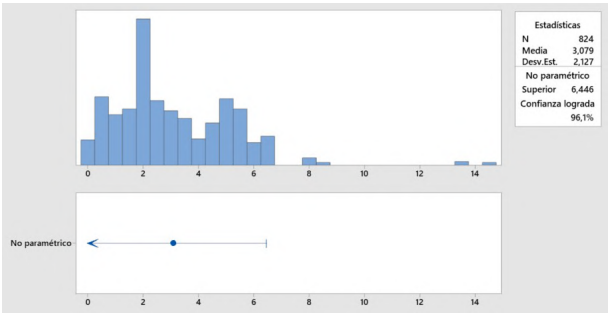


Figura 5.22: Intervalos de tolerancia no paramétricos del error en el Eje Yaw. Elaboración Propia: Minitab

las pruebas. En la ecuación 5.4 se presentan las de la posición, en donde la hipótesis nula corresponde cuando el error es mayor o igual a 10cm.

$$\text{Hipótesis: } \begin{cases} H_0 : \mu_{\text{error pos}} \geq 10 \text{ cm} \\ H_1 : \mu_{\text{error pos}} < 10 \text{ cm} \end{cases} \tag{5.4}$$

Por otro lado, en la ecuación 5.5, se expresa la hipótesis nula con un error de orientación mayor o igual a 15 grados.

$$\text{Hipótesis: } \begin{cases} H_0 : \mu_{\text{error orient}} \geq 15\ ^\circ \\ H_1 : \mu_{\text{error orient}} < 15\ ^\circ \end{cases} \tag{5.5}$$

A partir de ello, se obtienen los resultados de la tabla 5.4 en la cual la totalidad las pruebas presentó un valor de p de 0.000, lo cual indica que la evidencia estadística es lo suficientemente fuerte como para rechazar la hipótesis nula, en favor de la hipótesis alternativa. Es decir, los errores de posición mostrados de la secuencia de caminata se encuentran dentro de los márgenes establecidos por las especificaciones.

Tabla 5.4: Resultados de la prueba de hipótesis mediante la prueba de rango de Wilcoxon de una muestra

Variable de Prueba	Valor p
Error en el Eje X (m)	0.000
Error en el Eje Y (m)	0.000
Error en el Eje Z (m)	0.000
Error en Roll (°)	0.000
Error en Pitch (°)	0.000
Error en Yaw (°)	0.000

Carga computacional de la arquitectura

Asimismo, es importante analizar la carga del procesamiento cuando la arquitectura de control ejecuta la secuencia de caminata. En Fig. 5.23 se presenta el porcentaje de uso del procesador de la computadora central, en la cual se observa un consumo relativamente bajo, con valores de alrededor del 20% a lo largo del trayecto. Si bien esta secuencia de caminata no es de una gran complejidad, las trayectorias empleadas son las mismas que se emplean en algoritmos de control más complejos. En base a ello, se puede deducir que la carga actual es pequeña, por lo cual se espera un rendimiento adecuado al implementar módulos adicionales o algoritmos de control de mayor complejidad.

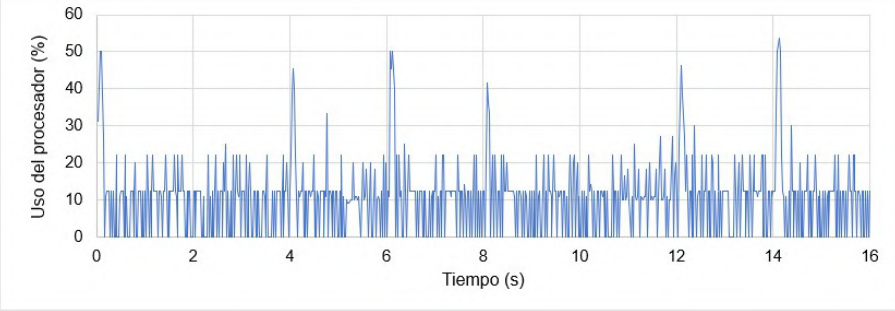


Figura 5.23: Porcentaje de uso del procesador durante la ejecución de la arquitectura en la secuencia de caminado. Elaboración Propia: Microsoft Excel

5.6 Implementación de un control de Torque PD

Durante estas pruebas, se observó que el robot experimentaba una serie de vibraciones y temblores en todo momento, desde el reposo hasta el balanceo de las piernas. En un inicio, el cliente había expresado el interés en rediseñar las piernas, debido a que, desde su punto de vista, las imperfecciones en las mismas causaban dichas sacudidas. Este proyecto se centra principalmente en el hardware y la programación del robot, por lo cual no se pretende realizar un estudio amplio de la situación. Sin embargo, se propone probar un algoritmo de control distinto al control de posición.

Inicialmente se empleó el control de posición interno de cada uno de los motores. El mismo consiste de un control PI, que recibe un ángulo deseado y una velocidad deseada. Con el fin de mostrar más a detalle la manera en que el robot empieza a temblar, se realiza una prueba, en donde se empuja ligeramente el robot y luego se observa su comportamiento. En Fig. 5.24 se puede observar las mediciones de los motores de una pierna durante dichas agitaciones. En ella se nota que los motores entran en un estado de inestabilidad, y continuarán oscilando a través del tiempo. A pesar de que la amplitud de estas oscilaciones parece pequeña, estas son capaces de ocasionar que el cuerpo del robot se mueva de un lado a otro, afectando considerablemente el desempeño de la locomoción, además de la percepción del cliente.

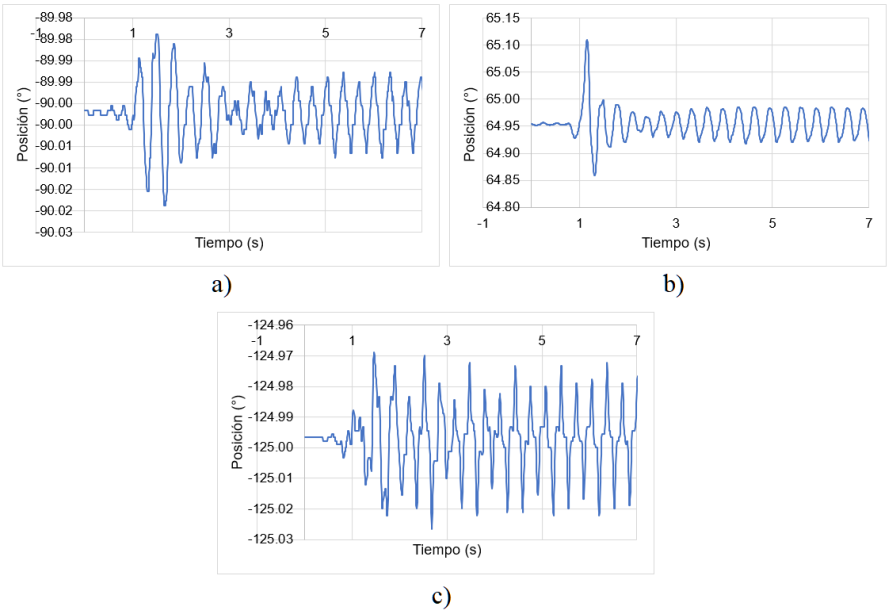


Figura 5.24: Comportamiento de la posición de las articulaciones ante una perturbación, empleando control de posición. a) Posición del hombro. b) Posición de la cadera. c) Posición de la rodilla. Elaboración Propia: Microsoft Excel

Dichas inestabilidades ocurren debido a que el control de posición es considerablemente rígido, sin embargo, existe un juego mecánico propio de los motores y además de la estructura del robot, por lo cual, se permite la propagación de perturbaciones en todo el cuerpo. Por otro lado, el control PI se puede modelar como resortes rotacionales en las

articulaciones, los cuales permiten el balanceo de un lado a otro.

Es por ello que se decide emplear un control PD en las articulaciones, con el fin de añadir un comportamiento de amortiguación a los motores y lograr atenuar dichas perturbaciones. Este control PD no ha sido desarrollado por simulaciones en las secuencias de caminado, sino que resulta de la obtención de parámetros de manera empírica, gracias a las capacidades de realizar pruebas rápidamente en la nueva arquitectura. Cabe recalcar que implementar este control en la secuencia de caminado resulta sumamente complejo, ya que se debe obedecer un comportamiento dinámico durante el balanceo y un correcto movimiento del centro de masa durante la fase del soporte. Es por ello que este control será implementado para ver su comportamiento en reposo, lo cual lo convierte en otra manera de medir la capacidad de la nueva arquitectura, en cuanto a lograr un comportamiento estable en un control de lazo cerrado de las 4 piernas.

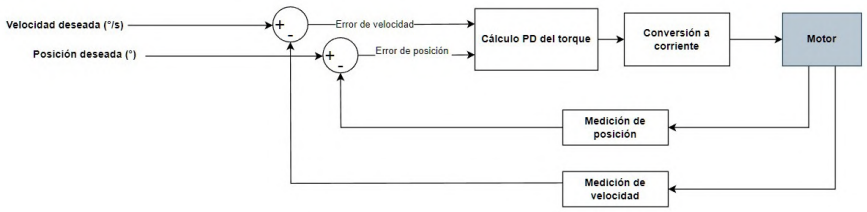


Figura 5.25: Diagrama del control PD de las articulaciones. Elaboración Propia: Microsoft Excel

En Fig. 5.25 se presenta el diagrama del control empleado, el cual posee una entrada de velocidad y posición deseadas y una salida del torque a los motores. Debido a que solo se pretende evaluar un comportamiento estático, se ajusta la velocidad deseada a 0. Los parámetros del control que mostraron un rendimiento satisfactorio para esta prueba corresponden a $[k_{p1}, k_{p2}, k_{p3}] = [55, 45, 75]$ y $[k_{d1}, k_{d2}, k_{d3}] = [1, 1, 1]$.

En Fig. 5.26 se observa el rendimiento de este control ante una perturbación similar a la realizada en el control de posición. En la misma se nota que el desplazamiento permitido es mucho mayor, sin embargo, se llega rápidamente al valor deseado y además, se logra una estabilidad. Asimismo, en la tabla 5.5 se presenta los resultados del estado transitorio del control, en donde se destacan tiempos de estabilización menores a 3 segundos y sobreimpulsos menores a 1°. Por otro lado, se obtuvo un pequeño error en la posición final, no obstante, el mismo es esperado al centrarse mayormente en el comportamiento resorte-amortiguador y la manera en que se comporta con las fuerzas y no en la posición.

A partir de ello se determina que los problemas de vibraciones del robot se pueden resolver significativamente mediante el empleo de otro control. Además, se valida la capacidad de la arquitectura de control para ejecutar un control con realimentación y lograr un estabilidad y comportamiento deseado.

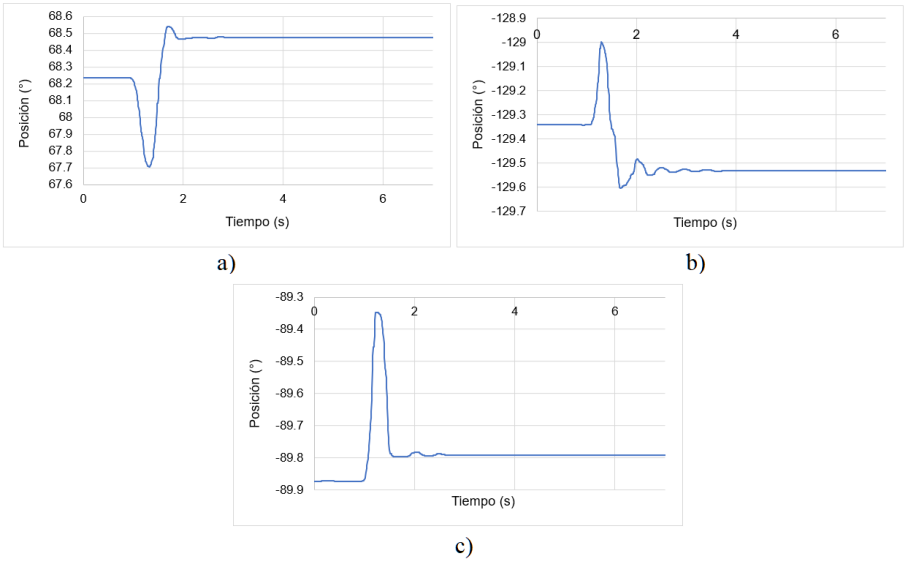


Figura 5.26: Comportamiento de la posición de las articulaciones ante una perturbación, empleando control de torque PD. a) Posición del hombro. b) Posición de la cadera. c) Posición de la rodilla. Elaboración Propia: Microsoft Excel

Tabla 5.5: Resultado del Control PD al nivel de las articulaciones

Parámetro	**Articulación**		
	Hombro	**Cadera**	**Rodilla**
Posición deseada (°)	-89.872	68.236	-129.340
Error permanente (°)	0.081	0.240	-0.191
Sobre-impulso máximo (°)	0.525	-0.528	0.340
Tiempo de estabilización (s)	1.63	0.675	2.77

Capítulo 6

Análisis Económico

En este apartado se realiza un análisis de los costos referentes al proyecto y la implementación de la solución, así como el retorno que se espera del producto en un futuro

6.0.1 Costos

Primeramente, se evalúan los costos materiales de la implementación física de la solución, correspondientes a la interfaz de comunicación y la interfaz de pruebas. En la tabla 6.1 se presenta el desglose de los mismos, en donde se observa que el componente de mayor peso en esto corresponde a los Módulos CAN USB, sin embargo, la totalidad de los mismos apenas alcanza $ 288.54, lo cual se encuentra en el rango de costos ideal del proyecto expresado previamente en las especificaciones. Estos componentes representan únicamente los que debieron ser comprados externamente. En la universidad se posee un taller que provee tanto componentes mecánicos como electrónicos, por lo cual se pueden acceder a ellos gratuitamente.

Componente	**Cantidad**	**Costo Unitario ($)**	**Costo Total ($)**
Módulo CAN USB	4	25.1	125.5
Adaptador USB - USB C	1	5.9	5.9
HUB USB	1	6.1	6.1
Botones cóncavos	5	3.2	16.2
LEDs Mixtos	1	10.6	10.6
Botón de emergencia	1	6.5	6.5
Relé	1	89.5	89.5
Filamento de Impresión 3D	1	29.3	28.3
	Total		288.5

Tabla 6.1: Costo de los componentes empleados en la solución

A partir del costo total del robot proporcionado en trabajos anteriores, el cual corres-

pondía a $ 6100 [30, p. 75], se determina que el nuevo costo total será de alrededor de $ 6400.

Por otro lado, se deben tomar en cuenta otros gastos como lo es la licencia necesaria de SolidWorks que se empleó para desarrollar la implementación física de los componentes, así como la actualización del modelo del robot. Asimismo, se presenta el costo del equipo de trabajo, en donde se empleó una computadora Lenovo Legion Y7000P con un valor de alrededor de $ 1200. Por último, se establece un costo salarial basado en un trabajo de medio tiempo en la universidad de estudio, el cual posee una remuneración de alrededor de $18 por hora. El resumen de esos costos se puede observar en la tabla 6.2 .

No.	**Detalle**	**Costo ($)**
1	Depreciación de equipos	100
2	Remuneración salarial	5760
3	Costo de licencia	49
	Total	5909

Tabla 6.2: Costos adicionales del desarrollo del proyecto

Adicionalmente, se compara la capacidad de la capa reactiva del robot, en tomando como referencia otros robots similares, en cuanto a tamaño y uso. El costo de estos robots está influenciado principalmente por los actuadores, en donde en el PAWDQ, por ejemplo, representan el 80% del costo total [28]. Es decir, la interfaz de comunicación no se ve reflejada de manera significativa en el precio de estos robots. No obstante, el presupuesto de cada uno de ellos también es congruente con los niveles de rendimiento esperados. En la tabla 6.3 se puede apreciar como la mayoría de ellos son capaces de alcanzar 1kHz en el muestreo de los datos, sin embargo, sus precios son relativamente elevados. Para el caso del Solo12, su precio resulta relativamente similar, pero esto se debe a que emplea actuadores mucho más pequeños. Es decir, el robot y el sistema diseñado presentan un precio competitivo en conjunto con un rendimiento bastante alto, el cual es capaz de cumplir con los requisitos propuestos por el cliente.

Robot	**Frecuencia de muestreo (Hz)**	**Costo** ($)
PAWDQ	1000	7700
DogoTix	500	12000
Mini Cheetah	1000	10000
Solo12 quadruped	1000	6300
AU Quadruped Robot	700	6400

Tabla 6.3: Comparación del rendimiento de la comunicación distintos robots cuadrúpedos

6.0.2 Beneficios del Proyecto

Debido a la naturaleza de investigación de este proyecto, no se espera que el mismo pueda retornar su costo mediante su funcionamiento, ya que este actualmente no desempeña aplicaciones industriales que le generen un propio beneficio económico al laboratorio. Por otro lado, este corresponde a las bases de una plataforma móvil para realizar distintas investigaciones, por lo cual, se espera que del mismo no se genere una utilidad económica, sino más bien la agilización para desarrollar proyectos en áreas como:

- Navegación autónoma en distintos terrenos.
- Desarrollo de secuencias de caminata.
- Implementación de simulaciones de alta fidelidad, o gemelos digitales.
- Desarrollo de aplicación con inteligencia artificial, como estrategias de control avanzadas.
- Fusión de sensores para aumentar la precisión de la lectura de variables en distintas situaciones.
- Visión por computación en aplicaciones específicas, como lo es en empleo de herramientas y manipulación de objetos.

Estos temas representan el interés principal de los miembros del laboratorio al completar este proyecto. Gracias al desarrollo de una arquitectura de control lo suficientemente veloz y flexible, se facilita el desarrollo e implementación física de los temas mencionados.

Estas investigaciones consisten en proyectos de desarrollo e investigación por parte de los estudiantes, así como tesis de maestría. Es decir, en los periodos lectivos consecutivos, se promoverá el apoyo de los estudiantes en los temas mencionados, tanto como el desarrollo del hardware y el software del robot. Prueba de ello corresponde que a la hora de finalizar este proyecto, actualmente un estudiante se encuentra iniciando sus tesis de maestría, la cual consiste en implementar un control MPC no lineal en el robot físico. Además, el mismo representa de un recurso educativo en cursos avanzados de maestría en la universidad, los cuales abarcan desde mecatrónica, robótica o incluso visión por computadora.

Por otro lado, al tener una plataforma robótica de mayor solidez, también se abren las puertas para que estudiantes de doctorado y post doctorado puedan desarrollar sus investigaciones en el mismo, lo cual se traduce en la publicación de múltiples artículos de investigación, ya sea en revistas o conferencias. Estos mismos aumentarán los indicadores de desempeño del laboratorio, así como su prestigio.

Asimismo, el laboratorio presenta convenios con otros departamentos de la universidad y con empresas exteriores, con el propósito de realizar investigaciones centradas en aplicaciones. Un ejemplo de ello consiste en un proyecto que surgió actualmente, el mismo

consiste en el monitorio continuo de humedales mediante un robot móvil autónomo que percibe los cambios en las emisiones de gases de efecto invernadero.

Estos acuerdos consisten en la inversión de grandes sumas de dinero en el laboratorio, y al presentar resultados tangibles, se abre las puertas para solicitar un mayor presupuesto. Es decir, mediante este proyecto se facilita las implementaciones físicas en el robot, lo cual representa un indicador sumamente positivo en estos proyectos de mayor calibre, y le permitirá al laboratorio obtener un mayor presupuesto en el desarrollo del mismo. Por lo tanto, esto da paso a la colaboración de más investigadores, un mayor número de publicaciones científicas, así como un aumento en la calidad del trabajo.

Capítulo 7

Conclusiones

7.1 Conclusiones

Al finalizar el desarrollo de este proyecto, se logró disminuir la brecha entre el robot físico y la simulación mediante una arquitectura que no solo contempla la funcionalidad necesaria, sino que también posee una gran flexibilidad para adaptar algoritmos de control rápidamente. Esto agilizará el desarrollo de investigaciones en esta plataforma en un futuro. Las principales conclusiones del proyecto son:

- Se determina que el robot real posee deficiencias en el rendimiento de su comunicación y su funcionalidad ralentiza la implementación de los algoritmos de control en base a las diferencias tan marcadas de la simulación, lo cual se refleja en una frecuencia de control de tan solo 64.8 Hz, en contraste con la expectativa inicial de 200 Hz.
- Se desarrolló una lógica para gestionar la comunicación con los 3 actuadores de cada una de las piernas, incrementando la frecuencia de muestreo a un 578.4% del valor original, esto empleando el hardware que se poseía en un inicio.
- Se desarrolla una arquitectura de control capaz de comunicarse con los 12 actuadores del robot a frecuencias mayores de 500 Hz y con una media de alrededor de 700 Hz, la cual se mantiene estable aun cuando se ejecutan algoritmos relativamente complejos.
- Se implementa una interfaz de control que permite un control absoluto en el robot real, lo cual logra emular las funcionalidades principales que se posee en el simulador, correspondientes al paro de emergencia, inicio del controlador, pausa el robot, reinicio de su posición y un comando extra.
- Se desarrolla una detección del contacto del suelo que deduce cuando una de las patas soporta una carga superior a 44.16 N, lo que permite su desempeño en secuencias de caminata que emplean hasta 3 patas en la fase de soporte, además de incluso detectar obstáculos en el camino.

- Se desarrolla una secuencia de caminata estáticamente estable, la cual es relativamente fiel a las simulaciones, con un error de posición promedio menor a 1.05 cm y un error de orientación promedio menor a 3.01 grados.

- Se comprueba el funcionamiento de la arquitectura en un control automático dependiente de la frecuencia de muestreo, mediante la implementación de un control PD de lazo cerrado al nivel de las articulaciones, el cual mostró un tiempo de respuesta ante perturbaciones menor a 3 s y con errores de posición menores a 1°,

- Se contribuyó con la estructuración de la arquitectura de control híbrida, la cual permite comportamientos reactivos, correspondientes al control de bajo nivel que funcionan altas velocidades, en conjunto con una capa ejecutiva que puede planear el movimiento del robot a una menor frecuencia.

7.2 Recomendaciones

- Se sugiere el empleo de la funcionalidad de tiempo real en la computadora central, la cual permite la disminución de la latencia, aumentando la frecuencia de comunicación y su estabilidad.

- Se recomiendo extender la detección de contacto a las consideraciones de cambios de orientación y aceleración del robot, con el fin de determinar si el soporte es de una calidad adecuada en términos de rigidez.

- Se aconseja el uso del simulador Gazebo, esto debido a que se tendrá una implementación directa con la arquitectura desarrollada.

- La secuencia de caminata puede tener una mayor velocidad de avance sin sacrificar su estabilidad significativamente, al planear un movimiento rápido del centro de masa para después ejecutar a su vez un balanceo de las piernas. De esta manera, se elimina el número de fases y se aumenta la fluidez del movimiento.

Bibliografía

[1] Unitree go1. https://shop.unitree.com/products, 2021. Consultado el 25 de enero de 2024.

[2] Raspberry pi 5. https://www.raspberrypi.com/products/raspberry-pi-5/, Oct. 2023.

[3] ArduPilot. Sitl with webots. https://ardupilot.org/dev/docs/sitl-with-webots.html. Accessed: January 14, 2024.

[4] S. Aunet. Nyquist rate analog-to-digital converters. Presented at Nanoelectronics Group, Dept. of Informatics, University of Oslo, NO, Mar. 2009.

[5] Marco Baglietto, Giorgio Battistelli, and Pietro Tesi. Packet loss detection in networked control systems. *International Journal of Robust and Nonlinear Control*, 30(15):6073–6090, 2020.

[6] G. Bledt, P. Wensing, S. Ingersoll, and S. Kim. Contact model fusion for event-based locomotion in unstructured terrains. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[7] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252, 2018.

[8] Bosch Sensortec. *BNO055 Intelligent 9-axis absolute orientation sensor Data sheet*, 2021. URL https://www.bosch-sensortec.com/products/smart-sensor-systems/bno055/#documents.

[9] Jennifer Chu. Mini cheetah is the first four-legged robot to do a backflip [online]. 2019 [visitado el 20 de marzo de 2024]. URL https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304.

[10] Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation and Koninklijke Philips Electronics N.V. Universal serial bus specification revision 2.0. Technical

report, Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V., 2000.

[11] J. A. Cook and J. Sj. Freudenberg. Controller area network (can). EECS 461, 2008.

[12] S. Corrigan. Introduction to the controller area network (can). Technical report, Texas Instruments, 2016.

[13] John J. Craig. *Robótica*. Pearson Educación, México D.F., 3rd edition, 2006.

[14] Cyberbotics. Simulating your robots with webots. https://cyberbotics.com/#webots. Accessed: January 14, 2024.

[15] Y. Du. Locomotion control scheme for quadruped robots, 2023.

[16] J. Faigl. Robotics paradigms and control arquitectures. Presented at B4M36UIR: Artificial Intelligence in Robotics, Czech Technical University, Prague, CZE, 2020.

[17] Thomas L. Floyd. *Dispositivos Electrónicos.* Pearson Educación, México D.F., 8th edition, 2008.

[18] Pablo Gonzalez and Elenda Garcia nad Joaquin Estremera. *Quadrupedal locomotion: an introduction to the control of four-legged robots*. Springer Science+Business Media, Madrid, España, primera edition, 2006.

[19] Grepow Inc., CA, EE.UU. *Tattu High Voltage Version*, 2022. URL https://www.genstattu.com/content/HV-LT-UAV.pdf.

[20] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Thomas Flayols, Jonathan Fiene, Alexander Badri-Spröwitz, and Ludovic Righetti. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020.

[21] Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST_NXP Wireless, and Texas Instruments. Universal serial bus 3.0 specification. Technical report, Intel Corporation, Compaq Computer Corporation, Hewlett-Packard Company, Royal Philips Electronics, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, 2008.

[22] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer. Anymal - toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931, 2017. URL https://doi.org/10.1080/01691864.2017.1378591.

[23] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer. Anymal - toward legged robots for harsh environments. *Advanced Robotics, 31:17, 918-931*, 2019.

[24] Marco Hutter, Christian Gehring, Michael Bloesch, Mark Hoepflinger, C Remy, and Roland Siegwart. Starleth: a compliant quadrupedal robot for fast, efficient, and versatile locomotion. pages 483–490, 09 2012.

[25] D. J. Hyun, J. Lee S. Seok, and S. Kim. High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah. *The International Journal of Robotics Research*, 2014. URL http://dx.doi.org/10.1177/0278364914532150.

[26] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros. Rl + model-based control: Using on-demand optimal control to learn versatile legged locomotion. https://doi.org/10.1109/LRA.2023.3307008, 2023.

[27] B. Katz1, J. Di Carlo, and S. Kim1. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. *International Conference on Robotics and Automation (ICRA)*, 2019.

[28] J. Kim, T. Kang, D. Song, and S.-J. Yi. Design and control of an open-source, low cost, 3d printed dynamic quadruped robot. *Applied Sciences*, 11:3762, 2021. URL https://doi.org/10.3390/app11093762.

[29] F. Martín. *Robot Programming with ROS2*. CRC Press, Florida, EE.UU., 1st edition, 2023.

[30] S. Nørregaard and S. Thorenfeldt. Design and development of a four legged robot. Technical report, Aarhus University, 2022.

[31] S. Nørregaard and S. Thorenfeldt. Towards a kinematic and dynamic control system for a four-legged robot, 2022.

[32] Raspberry Pi, Cambridge, UK. *Raspberry Pi 4 Model B Datasheet*, 2019. URL https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf.

[33] J. Ren, Y. Dai, B. Liu, P. Xie, and G. Wang. Hierarchical vision navigation system for quadruped robots with foothold adaptation learning. *Sensors*, 23:5194, 2023. URL https://doi.org/10.3390/s23115194.

[34] A. Román. I2c serial protocol. https://medium.com/@alexjromnmuiz/i2c-serial-protocol-1c42aee8ebd9, 2021.

[35] Dirk Thomas. Changes between ROS 1 and ROS 2. https://design.ros2.org/articles/changes.html, Sep. 2015. Last Modified: Jun. 2017.

[36] D. Touretzky and E. Tira-Thompson. Architectures for robot control. Presented at 15-494: Cognitive Robotics, Carnegie Mellon University, Pensilvania, USA, Feb. 2008.

[37] K. T. Ulrich and S. D. Eppinger. *Diseño y desarrollo de productos.* McGraw-Hill/Interamericana Editores S.A. de C.V., México D.F., quinta edition, 2013.

[38] J. Valdez and J. Becker. Understanding the i2c bus. Technical report, Texas Instrumens, 2015.

[39] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, and Keying Ye. *Probabilidad y estadística para ingeniería y ciencias.* Pearson Educación, México, novena edition, 2012. Área: Ingeniería.

[40] L. Wan. Model-based control strategy for a quadruped robot. Technical report, Aarhus University, 2023.

[41] Xuping Zhang. Robotics & control [online]. 2023 [visitado el 20 de marzo de 2024]. URL https://mpe.au.dk/forskning/faciliteter/robotics.

[42] A. A. Zheltoukhov and L. A. Stankevich. A survey of control architectures for autonomous mobile robots. *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2017.

Apéndice A

Bitácora de diseño

A.1 Metodología

A.1.1 Obtención de las necesidades del proyecto

Los comentarios más importantes de la entrevista, de los cuales se derivan las necesidades planteadas del proyecto se presentan en la tabla A.1, los mismos se tradujeron del inglés.

Tabla A.1: Necesidades interpretadas a partir de la entrevista

Cliente	Xuping Zhang	
Fecha	22 de septiembre, 2023	
Pregunta	**Enunciado del cliente**	**Necesidad Interpretada**
Producto Final	"Me gustaría que el robot sea capaz de al menos caminar alrededor del edificio".	El robot puede caminar una distancia considerable.
Añadir componentes	"En futuro se integrarán nuevos componentes al robot".	El SD posee un diseño modular.
Algortimos de control	"Anteriormente se han trabajado en los controles MPC y VMC, entre más simples mejor".	El sistema diseñado (SD) puede manejar algoritmos de control complejos.
Instrucciones de la computadora central	"Se precisa el control de movimiento pero también se pueden incluir interacciones con el ambiente".	El SD es capaz de procesar todos los datos. El bucle de control del robot se mantiene estable.
Facilidad de control	"Quiero que se pueda controlar tan fácil como una persona usa un celular, de manera ideal".	El sistema es fácilmente controlable por el usuario.
Tipo de control	"Debe ser al menos semiautomático, que al apretar un botón el robot pueda caminar".	El robot realiza tareas complejas con instrucciones simples.
Velocidad del hardware	"Que este sea lo suficientemente veloz para procesar los sensores, actuadores y el algoritmo de control".	El sistema diseñado lee los sensores correctamente.
Seguridad entre humanos y robots	"Creo que la seguridad debe tomarse en cuenta, es la máxima prioridad".	El robot realiza movimientos seguros para los humanos.
Importancia de la recepción de mensajes	"Se debe proveer suficiente información al algoritmo de control".	El SD le proporciona la información necesaria al controlador en el momento apropiado.
Prioridad en las funciones	"La seguridad debe ser primero, luego un movimiento confiable".	Las funciones del robot respetan una jerarquía de importancia.
Facilidad de implementación de algoritmos de control	"Si los operarios tuvieran una interfaz sería lo ideal, pero este debería ser capaz de implementarlo directamente en el código".	El sistema permite una fácil implementación de algoritmos de control.
Integración de componentes	"Me gustaría que el diseño fuera modular".	El SD posee un diseño modular.
Secuencias de caminata	"Se pueden tomar en cuenta las investigadas en trabajos anteriores, pero el principal objetivo es que logre caminar".	El SD puede ejecutar diferentes secuencias de caminado.
Terrenos objetivo	"Solo se requiere que pueda caminar sobre el concreto y bajo techo".	El robot puede moverse a través de varios tipos de terrenos.
Estabilidad de movimientos	"Los errores no son sumamente importantes, que al menos pueda caminar, sin embargo, se debe notar un cierto nivel de estabilidad".	El cuerpo del robot permanece estable durante los movimientos.
Semejanza de la simulación al robot físico	"Debería tener una desviación máxima de un 15%".	El robot físico se asemeja estrechamente al comportamiento cinemático de la simulación.
Balanceo de las piernas	"Debería ser capaz de moverse y además que sea lo más natural posible, inspirado en la biomecánica".	El robot realiza movimientos fluidos.
Velocidad de movimiento	"Me gustaría que el robot se pudiera mover tan rápido como un perro cachorro caminando".	El robot se desplaza a una velocidad considerable.
Facilidad de modificación de la arquitectura	"La arquitectura de control debe ser fácilmente modificable, que sea lo más flexible posible".	La arquitectura de control del SD se puede modificar fácilmente.
Consumo de poder	"El robot debe ser capaz de moverse 45 minutos con una carga completa o incluso 1 hora".	El robot opera continuamente durante un período significativo.
Presupuesto	"Se dispone de alrededor de 3000 dkk para este proyecto, pero sí se propone algo que lo amerite, se puede destinar un mayor presupuesto".	El SD tiene un costo adecuado.
Tiempo de desarrollo	"Me gustaría ver el robot caminando después de 5 meses".	El SD se desarrolla dentro de un marco de tiempo óptimo.
Calibración	"La calibración debe ser fácil de realizarse, pero que no se necesite frecuentemente, alrededor de cada 6 meses".	El robot requiere calibración en intervalos óptimos.

A.1.2 Valores objetivo del proyecto

Tabla A.2: Tabla de Métricas y Valores Objetivo

Núm.	Métrica	Imp.	Unidad	V. Marginal	V. Ideal
1	Algoritmos de control compatibles	4	Lista	- Cinemático - VMC - MPC	- Cinemático - VMC - MPC - WBC - Con IA
2	Facilidad de Interacción	5	Subjetiva	> 3	5
3	Tipo de funcionamiento	4	Lista	- Semiautomático	- Semiautomático - Automático
4	Porcentaje de mediciones no recibidas	3	%	< 3	$\ll 1$
5	Monitoreo de parámetros	4	Lista	- Temperaturas - Posiciones - Velocidades - Fuerzas	- Temperaturas - Posiciones - Velocidades - Fuerzas
6	Funciones de Seguridad	4	Lista	Paro de emergencia	- Paro de emergencia - Detección de humanos
7	Frecuencia de muestreo	4	Hz	> 100	> 500
8	Frecuencia del lazo de control de nivel alto	3	Hz	> 50	> 100
9	Jerarquía de Funciones	4	Lista	1. Seguridad 2. Movimientos 3. Funciones Extra	1. Seguridad 2. Movimientos 3. Funciones Extra
10	Facilidad de Implementación	4	Subjetiva	> 3	> 4
11	Componentes que se pueden añadir	4	Lista	- Lidar - Cámara	- Lidar - Brazo Robótico - Herramientas
12	Secuencias de caminata	5	Lista	Gateo	- Gateo - Trote - Giro
13	Tipos de terrenos	4	Lista	Planos	Planos
14	Distancia de avance	5	m	>25	>75
15	Porcentaje de similitud de la simulación	4	%	>80	>90
16	Trayectorias	4	Lista	Cicloidal	- Cicloidal - Bézier
17	Velocidad de movimiento	4	m/s	> 0.25	> 0.5
18	Facilidad de implementación	5	Subjetiva	> 3	>5
19	Periodo de funcionamiento	4	min	>40	> 60
20	Error de Orientación	5	$grados$	< 20	< 15
21	Error de Posición	5	cm	< 10	< 5
22	Periodo de Calibración	3	$meses$	< 6	< 1
23	Porcentaje de variación del muestreo	5	%	< 20	< 10
24	Costo	5	usd	< 600	< 400
25	Tiempo de desarrollo	4	$meses$	< 6	< 5

Tabla A.3: Comentarios de los valores objetivo

Número	Métrica	Comentario
1	Algoritmos compatibles	Controles implementados en trabajos anteriores y los propuestos para el futuro.
2	Facilidad de interacción	Encuesta del 1 al 5 que refleja la interpretación del uso y la respuesta del robot con el nuevo sistema.
3	Tipo de funcionamiento	Tipo de funcionamiento de las secuencias de caminata.
4	Porcentaje de mediciones no procesadas	Errores admisibles en la recepción de paquetes para un sistema de control[5].
5	Monitoreo de parámetros	Parámetros relevantes de los componentes del robot.
6	Funciones de seguridad	Variables que detectan el impacto o atascamiento con el entorno.
7	Frecuencia de muestreo	Valores de frecuencia requeridos por los controles actuales y existentes.
8	Funciones	Lista de prioridades establecidas por el cliente.
9	Frecuencia del lazo de control	Frecuencia del lazo de control de alto nivel del MPC.
10	Facilidad de implementación	Encuesta del 1 al 5 que refleje la facilidad de llevar un código de simulación a la arquitectura.
11	Componentes que se pueden añadir	Lista de componentes considerando investigaciones futuras.
12	Secuencias de caminata	Secuencias estudiadas en trabajos anteriores.
13	Tipos de terreno	Tipo de terreno establecido por el cliente.
14	Distancia máxima	Distancia a recorrer establecida por el cliente.
15	Porcentaje de similitud	Porcentaje de similitud del movimiento del centro de masa respecto a la simulación establecida por el cliente.
16	Trayectorias	Trayectorias empleadas en trabajos anteriores y en otros robots cuadrúpedos.
17	Velocidad de movimiento	Velocidad similar a la de un perro cachorro caminando, según referencia del cliente.
18	Facilidad de modificación	Encuesta del 1 al 5 que refleje la facilidad de modificar módulos en la arquitectura de control.
19	Periodo de funcionamiento	Periodo de funcionamiento esperado por el cliente.
20	Error de orientación	Error de orientación basado en resultados de simulaciones anteriores.
21	Periodo de calibración	Periodo de calibración esperado por el cliente.
22	Porcentaje de variación	Porcentaje de variación en la comunicación basado en resultados iniciales.
23	Costo	Presupuesto del proyecto establecido por el cliente.
24	Tiempo de desarrollo	Tiempo de desarrollo establecido por el cliente.

A.1.3 Búsqueda de Información

En esta sección se presenta un análisis comparativo de las distintas soluciones a la comunicación de los actuadores del robot, así como las generalidades de una de las curvas de balanceo, las cuales se emplearon en el desarrollo de la trayectoria.

Sistemas de comunicación

En la tabla A.4 se presenta una comparación de las interfaces de hardware empleadas en distintos robots cuadrúpedos que resultan similares al robot con el que se trabajó, asimismo, se especifica el rendimiento de las mismas para ejecutar un control de bajo nivel.

Robot	**Comunicación**	**Frec. (Hz)**	Fuente
MIT MiniCheetah	Tarjeta propia de 4 canales CAN que se comunica mediante SPI	1000	[27]
MIT Cheetah 3	Tarjeta propia de 4 canales CAN que se comunica mediante EtherCAT	1000	[7]
PADWQ	4 Módulos CANable USB-CAN	1000	[28]
StarlETH	4 redes CAN (desconocido)	400	[24]
ANYmal	4 redes CAN (desconocido)	400	[22]
DOGOTIX (Tsinghua University)	Tarjeta propia de 2 canales CAN que se comunica mediante SPI	500	[33]
Solo12 quadruped	Microcontrolador máster con 4 módulos CAN, el cual se comunica por Ethernet	1000	[20]

Tabla A.4: Análisis comparativo de la comunicación de otros robots cuadrúpedos

Generalidades de la Curva de Bézier

La curva de Bézier consiste de una trayectoria empleada comúnmente en robot móviles articulados. Esta permite mantener un suficiente espacio para evitar obstáculos durante la trayectoria de la pierna, además, permite controlar la velocidad a la que se retrae y extiende la pierna, lo cual resulta en una reducción de las pérdidas de energías al hacer contacto con el suelo. Esto debido a que posibles impactos al final de la trayectoria cuando el robot se está moviendo, frenarán su avance, además de poder desviar ligeramente su recorrido.

En este trabajo, se empleará una curva de Bézier previamente diseñada por el MIT, para su robot el Cheetah 3. Esta consiste de 12 puntos de control fijos, mostrados en la tabla

A.5. Esta trayectoria se puede ver en Fig. A.1, la cual consta de 3 fases, correspondientes a seguimiento, protracción y retracción.

Punto	X (mm)	Y(mm)
c0	-200.0	500.0
c1	-280.5	500.0
c2	-300.0	361.1
c3	-300.0	361.1
c4	-300.0	361.1
c5	0.0	361.1
c6	0.0	361.1
c7	0.0	321.4
c8	303.2	321.4
c9	303.2	321.4
c10	282.6	500.0
c11	200.0	500.0

Tabla A.5: Puntos de Control para la curva de Bézier. Recuperado de: [25]

Los puntos de control actúan como anclas de múltiples curvas. A partir de ellos se calcula la posición en cada una de las coordenadas, en base a la ecuación A.1, en donde $B_k^n(t)$ corresponde al polinomio de Bernstein de grado n y c_k corresponde al punto de control k. Esto se calcula para ambos ejes X y Y, en donde el polinomio de Bernstein se mantiene igual para ambos casos, sin embargo, se utiliza la componente del punto de control correspondiente.

$$p(t) = \sum_{k=0}^{n} c_k B_k^n(t) \tag{A.1}$$

El posicionamiento de los puntos de control consiste en la principal funcionalidad de esta curva para diseñar trayectorias. Para este caso en específico, se destacan las siguientes características recuperadas de [25]:

- Se pueden solapar las coordenadas de dos puntos de control para indicar velocidad cero en alguno de los ejes.
- Se solapan 3 puntos de control al cambiar la dirección de avance para establecer una aceleración nula.
- Se emplean dos pares de puntos de control solapados para cambiar la dirección de la trayectoria durante la protracción de la pierna
- Se solapan dos puntos de control en el final de la trayectoria para establecer una velocidad cero en el eje vertical al tocar el suelo, disminuyendo el impacto.

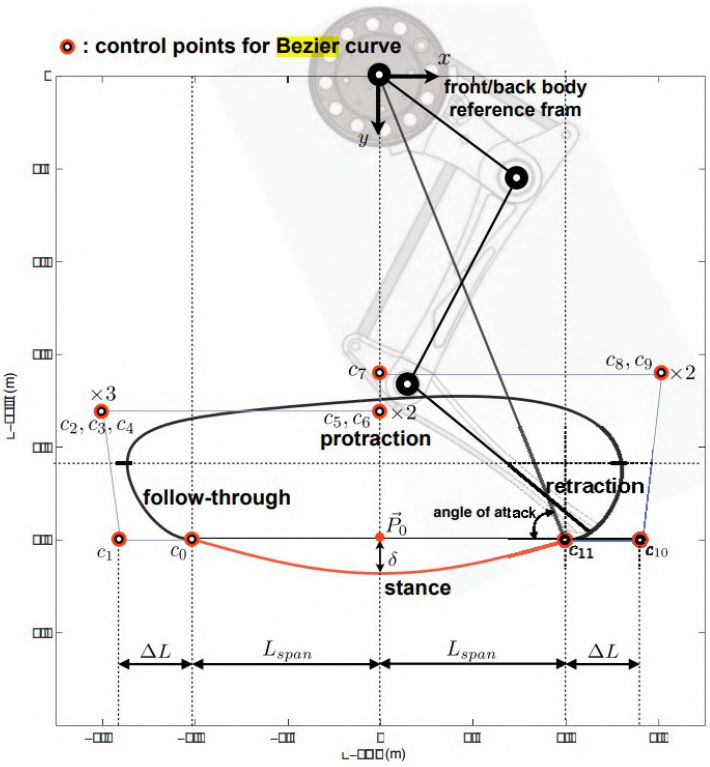


Figura A.1: Curva de Bezier de referencia. Recuperado de: [25]

A.2 Consideraciones adicionales de seguridad

A.2.1 Medición del porcentaje de la batería

Para determinar el porcentaje de carga de una batería de polímero de litio, se debe emplear la curva de descarga específica del modelo empleado. El mismo corresponde a "TATTU 11000MAH 22.8V HV 25C 6S1P", para el cual se toma la curva de 4.2V como referencia en [19]. A partir de ello, se trazan puntos de interés hasta lograr una aproximación considerable, los mimos se encuentran en la tabla A.6. En base a esto se traza una curva formada por funciones lineales, tal como se muestra en Fig. A.2. De esta manera, se obtienen las funciones lineales que dependen del voltaje de la batería, y se podrá estimar el porcentaje de carga actual.

Tabla A.6: Puntos para la gráfica del porcentaje de carga de la batería

N. punto	Voltaje	Porcentaje (%)
1	4.17	100
2	4.07	90
3	3.96	80
4	3.905	70
5	3.84	60
6	3.8	50
7	3.775	40
8	3.75	30
9	3.715	20
10	3.67	10
11	3.66	7.5
12	3.62	5
13	3.47	2.5
14	3	0

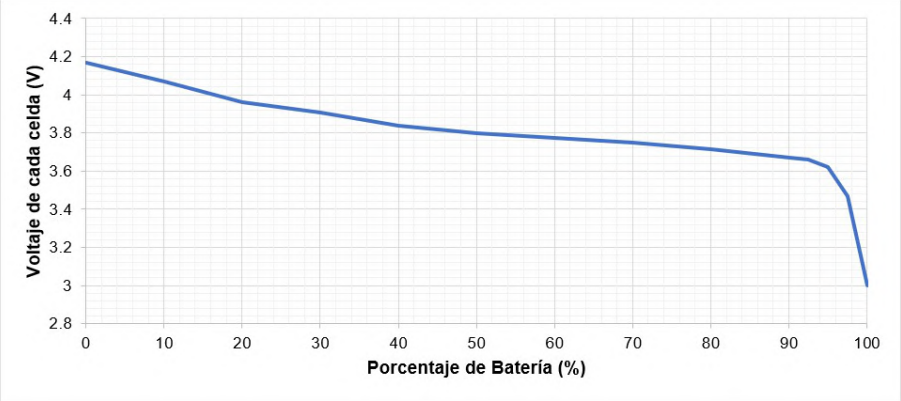


Figura A.2: Linealización de la curva de descarga de la batería. Elaboración propia: Microsoft Excel

A.2.2 Especificaciones Finales

Tabla A.7: Tabla de especificaciones finales

Núm.	Métrica	Imp.	Unidad	V. Final
1	Algoritmos de control compatibles	4	Lista	- Cinemático - VMC - MPC
2	Facilidad de Interacción	5	Subjetiva	4
3	Tipo de funcionamiento	4	Lista	Semiautomático
4	Porcentaje de mediciones no recibidas	3	%	0
5	Monitoreo de parámetros	4	Lista	- Temperaturas - Posiciones - Torques - Voltaje de la batería
6	Funciones de Seguridad	4	Lista	Paro de emergencia
7	Frecuencia de muestreo	4	Hz	700
8	Frecuencia del lazo de control de nivel alto	3	Hz	100
9	Jerarquía de Funciones	4	Lista	1. Seguridad 2. Movimientos 3. Funciones Extra
10	Facilidad de Implementación	4	Subjetiva	4
11	Componentes que se pueden añadir	4	Lista	- Lidar - Cámara
12	Secuencias de caminata	5	Lista	Gateo
13	Tipos de terrenos	4	Lista	Planos
14	Distancia de avance	5	m	200
15	Porcentaje de similitud de la simulación	4	%	>85
16	Trayectorias	4	Lista	Bézier
17	Velocidad de movimiento	4	m/s	>0.013
18	Facilidad de implementación	5	Subjetiva	4
19	Periodo de funcionamiento	4	min	30
20	Error de Orientación	5	$grados$	6.45
21	Error de Posición	5	cm	< 2.3
22	Periodo de Calibración	3	$meses$	1
23	Porcentaje de variación del muestreo	5	%	< 12
24	Costo	5	usd	290
25	Tiempo de desarrollo	4	$meses$	6

A.3 Planos Eléctricos

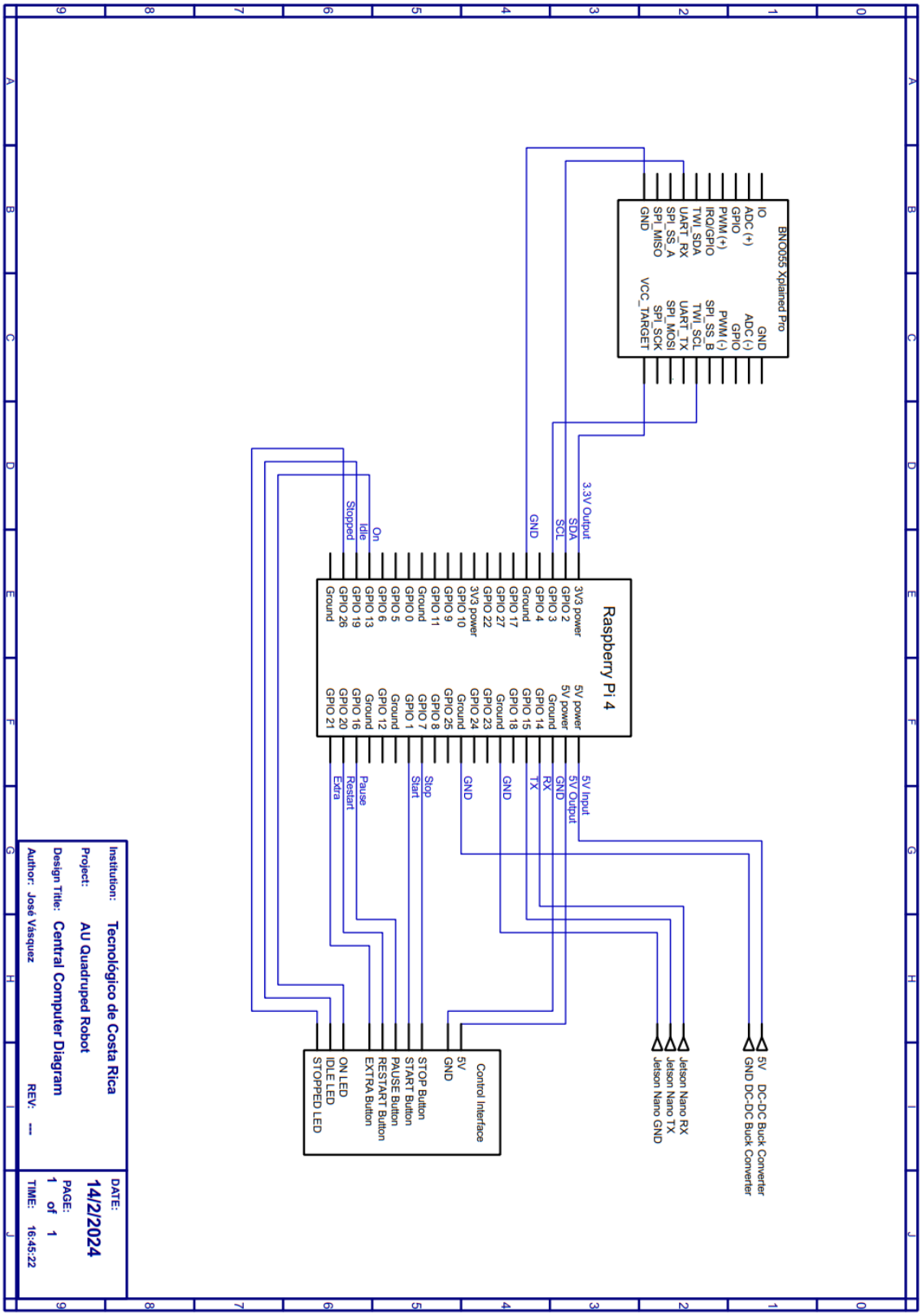


Figura A.3: Plano eléctrico de la computadora central. Elaboración Propia: Proteus

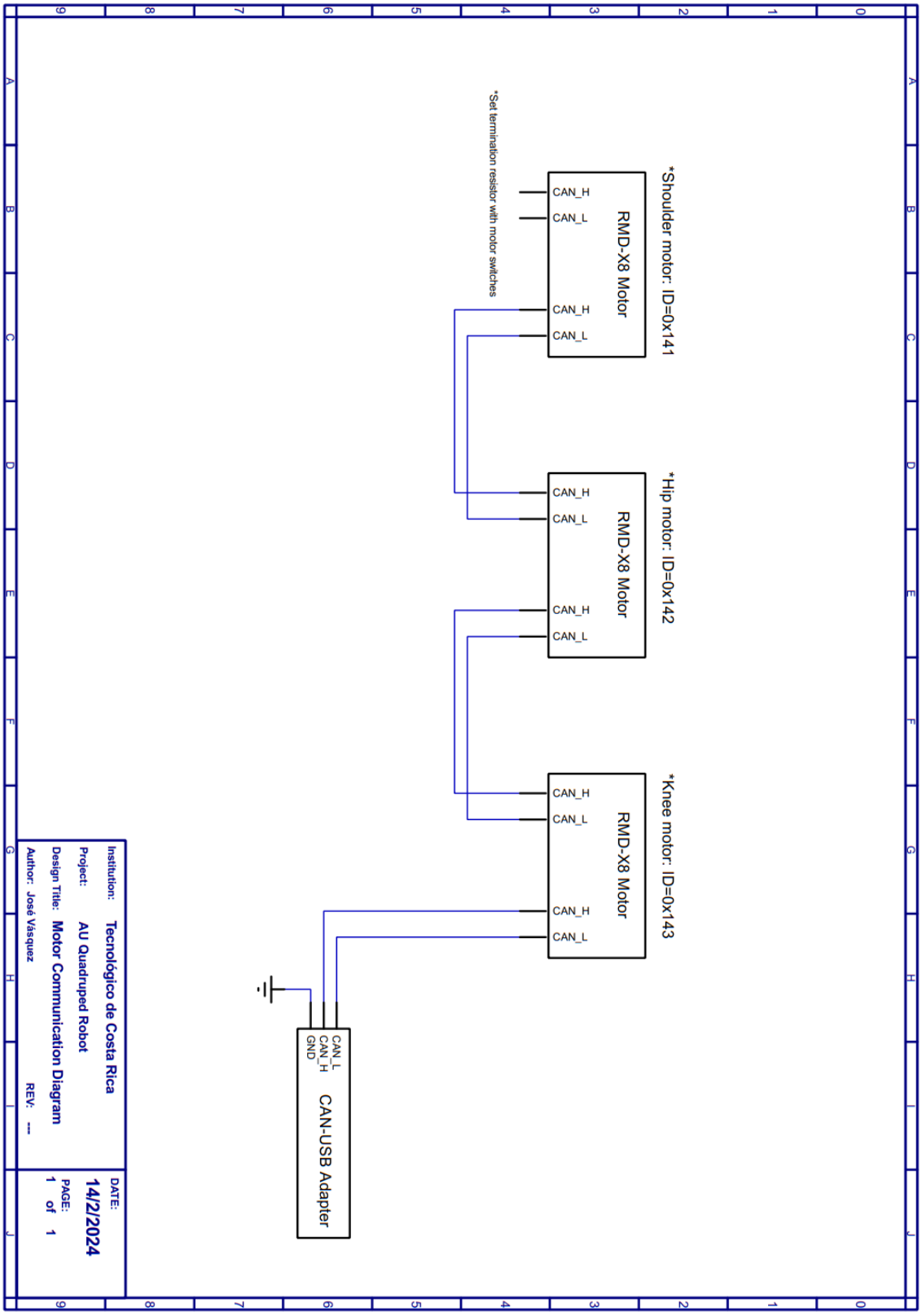


Figura A.4: Plano eléctrico de la comunicación de los motores de cada pierna. Elaboración Propia: Proteus

Apéndice B

Experimentos

B.1 Datos de los experimentos realizados

En esta sección se presenta un complemento a las pruebas del movimiento del centro del robot. En el análisis estadístico, se empleó un ciclo completo de la secuencia de caminata, correspondiente a alrededor de 800 muestras para el análisis. En la tabla B.1 se muestra el promedio de error de 30 ciclos de caminata, los cuales mostraron ser consistentes a los resultados de la prueba 1, es decir, esta constituyó un buen referente para realizar la estadística.

Tabla B.1: Promedio de los errores de posicionamiento y orientación del centro de masa del robot

Núm. Prueba	**Eje X (m)**	**Eje Y (m)**	**Eje Z (m)**	**Roll** (°)	**Pitch** (°)	**Yaw** (°)
1	0.0103	0.0051	0.0614	0.3554	2.4972	3.0790
2	0.0102	0.0051	0.0611	0.3779	2.4084	3.1169
3	0.0097	0.0050	0.0610	0.3735	2.7411	2.9076
4	0.0100	0.0053	0.0612	0.5188	2.4568	2.9786
5	0.0099	0.0054	0.0611	0.3642	2.4900	2.9336
6	0.0102	0.0052	0.0612	0.4900	2.8272	3.1082
7	0.0098	0.0053	0.0608	0.4854	2.7072	3.3980
8	0.0098	0.0053	0.0608	0.4282	2.3681	5.2958
9	0.0097	0.0052	0.0605	0.9427	5.0034	4.1204
10	0.0101	0.0054	0.0609	0.4914	3.0306	3.1417
11	0.0100	0.0048	0.0610	0.3912	2.4443	2.8755
12	0.0098	0.0046	0.0611	0.2737	2.2975	3.0596
13	0.0102	0.0048	0.0612	0.3954	2.3556	3.0514
14	0.0098	0.0053	0.0607	0.4630	2.8234	3.2553
15	0.0096	0.0054	0.0605	0.7965	2.8715	3.4309
16	0.0098	0.0054	0.0609	0.4448	2.6045	3.1853
17	0.0102	0.0053	0.0611	0.3415	2.4398	2.7344
18	0.0093	0.0049	0.0606	0.4148	2.6368	3.4283
19	0.0105	0.0055	0.0613	0.2536	2.4680	2.9762
20	0.0103	0.0051	0.0613	0.3549	2.5201	3.4137
21	0.0101	0.0052	0.0612	0.4524	2.7211	3.5151
22	0.0102	0.0050	0.0613	0.4325	2.3647	2.9892
23	0.0100	0.0051	0.0613	0.3340	2.1613	2.9986
24	0.0099	0.0051	0.0609	0.4835	2.7562	3.5496
25	0.0102	0.0053	0.0612	0.3519	2.3565	1.9790
26	0.0097	0.0052	0.0606	0.7216	2.6167	3.3140
27	0.0102	0.0047	0.0614	0.3234	2.3087	3.0629
28	0.0103	0.0051	0.0613	0.4169	1.9804	2.4757
29	0.0100	0.0051	0.0610	0.3920	2.2876	2.9759
30	0.0097	0.0051	0.0608	0.5240	2.6141	2.8607
Promedio:	0.0099	0.0055	0.0573	0.4138	2.397	3.0305