UNIVERSIDAD DE TARAPACÁ

UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Ingeniería@
Computación e Informática

Proyecto “Circuitron”
Robot Bipedo Humanoide

Integrantes: Brayan Garcia
Pablo Varas

Profesor: Diego Aracena Pizarro

Curso: Proyecto 2

ARICA, Lunes 4 NOVIEMBRE 2024

Índice

Panorama General

Resumen del proyecto

Introducción

El proyecto "Circuitron" tiene como objetivo principal la creación de un robot bípedo humanoide diseñado para caminar de manera eficiente, estable y autónoma. Este desafío implica abordar aspectos complejos relacionados con la mecánica, la electrónica y la programación, integrando tecnologías avanzadas para garantizar que el robot logre desplazarse imitando el movimiento humano. El enfoque del proyecto incluye el diseño y fabricación de la estructura del robot, la implementación de sistemas de control para la estabilidad y el equilibrio, y el desarrollo de algoritmos que permitan movimientos fluidos y coordinados. A través de este proyecto, se busca avanzar en el desarrollo de tecnologías aplicadas a la robótica, con énfasis en el perfeccionamiento de sistemas de locomoción humanoide.

Problemática

El desarrollo de robots bípedos que puedan caminar de manera eficiente y equilibrada es uno de los mayores desafíos en robótica. A diferencia de los sistemas con ruedas o múltiples patas, los robots humanoides deben mantener el equilibrio dinámico en diversas situaciones, como caminar en terrenos irregulares, subir escaleras o cambiar de dirección abruptamente. Esto implica un control preciso de múltiples grados de libertad, respuesta rápida a perturbaciones externas, y el desarrollo de algoritmos avanzados de control para la estabilidad.

Aspectos a considerar en el informe:

- **Desafíos mecánicos**: Diseño de las articulaciones y distribución de peso para optimizar el centro de masa.
- **Desafíos de control**: Uso de sensores como giroscopios y acelerómetros para mantener el equilibrio en tiempo real.
- **Desafíos de software**: Implementación de algoritmos de control como el control de retroalimentación, redes neuronales o aprendizaje por refuerzo.

Solución

Para abordar los desafíos de estabilidad y control de movimiento en el robot bípedo, se propone implementar un sistema de control de equilibrio basado en retroalimentación de sensores, combinado con algoritmos predictivos que anticipen y ajusten las acciones del robot en función del entorno y su postura actual.

Ventajas de la solución:

- **Estabilidad mejorada**: La retroalimentación en tiempo real y los algoritmos predictivos mejorarán la capacidad del robot para mantener el equilibrio en situaciones dinámicas.
- **Optimización continua**: A través de la simulación y la retroalimentación de datos, se podrá refinar el control de movimiento, adaptando el robot a diferentes entornos sin tener que modificar el hardware.

Esquema Solución

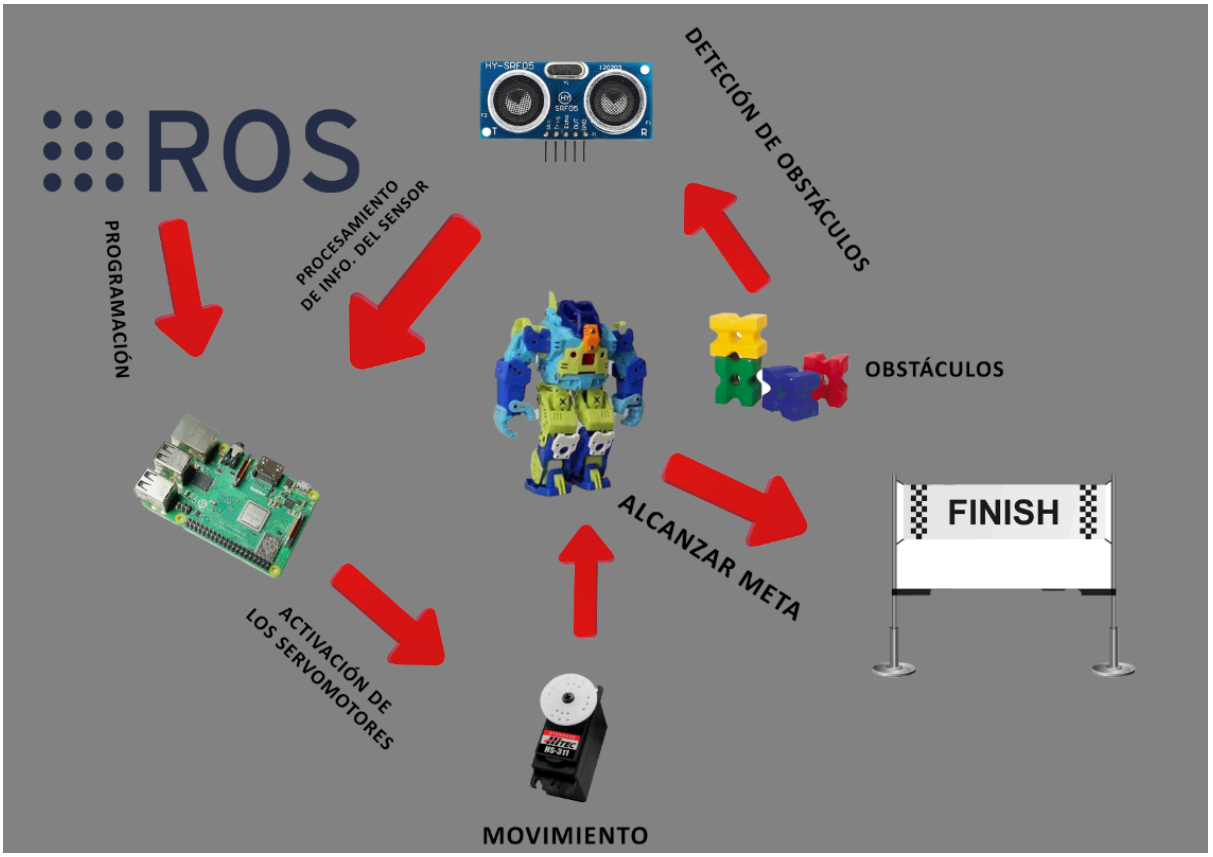

fig n°1: Esquema que modela el funcionamiento del robot.

Alcance

A través de la participación de nuestro robot en la competencia Ingeniería 2030, buscamos fomentar la participación activa de los estudiantes y generar un impacto positivo en la promoción de la robótica como disciplina educativa.

Objetivo General

Desarrollar un robot bípedo humanoide capaz de desplazarse de manera autónoma y eficiente a través de circuitos con obstáculos y desniveles, integrando tecnologías avanzadas de control y detección que le permitan adaptarse dinámicamente a diferentes entornos y superar los desafíos de equilibrio y movilidad.

Objetivo Específicos

- Desarrollar un sistema de locomoción eficiente que permita al robot bípedo desplazarse de manera autónoma y mantener el equilibrio en terrenos irregulares y cambiantes.
- Diseñar una interfaz de usuario que permita monitorear en tiempo real el estado del robot y realizar ajustes durante la competencia.
- Implementar un sistema de detección y reconocimiento de obstáculos utilizando sensores como sensor ultrasonido.
- Establecer un sistema de control de movimientos articulados que permita al robot superar diferentes tipos de obstáculos.
- Realizar pruebas y ajustes iterativos del hardware y software para optimizar el rendimiento del robot.

Suposiciones y Restricciones

Suposiciones

Entorno controlado: Se asume que el robot operará inicialmente en un entorno controlado y predefinido (superficies planas, sin obstáculos imprevistos), lo que facilita el desarrollo y pruebas iniciales de la locomoción bípeda.

Disponibilidad de tecnología: Se presupone que se cuenta con acceso a sensores avanzados y sistemas de control moderno para la ejecución de algoritmos en tiempo real.

Materiales adecuados: Se asume que los materiales utilizados para la estructura del robot serán ligeros y resistentes (como aleaciones de aluminio, fibra de carbono o plásticos de alta resistencia), lo que permitirá la reducción de peso sin comprometer la durabilidad.

Fuente de energía eficiente: Se supone que el robot estará equipado con baterías recargables de alta capacidad, suficientes para asegurar un funcionamiento continuo durante pruebas y demostraciones prolongadas.

Colaboración interdisciplinaria: Se asume que se contará con la colaboración de ingenieros especializados en diferentes áreas (mecánica, electrónica) para abordar cada fase del proyecto.

Restricciones

Limitaciones de energía: Aunque se prevé una fuente de energía eficiente, la capacidad de la batería restringe la autonomía del robot, limitando la duración y la complejidad de las pruebas de locomoción.

Complejidad en la movilidad: El robot enfrenta restricciones en cuanto a su capacidad para desplazarse en terrenos irregulares o con pendientes pronunciadas, debido a la complejidad de equilibrar el movimiento bípeda sin caídas.

Tiempo de desarrollo: Se enfrenta una restricción temporal en el desarrollo del proyecto, lo que implica que algunos aspectos avanzados de la locomoción y el control de movimientos deberán simplificarse o postergarse para futuras iteraciones.

Presupuesto limitado: El desarrollo del robot debe ajustarse a un presupuesto limitado, lo que restringe la adquisición de componentes más avanzados o soluciones de hardware/software de alto costo.

Peso máximo permitido: La estructura del robot deberá ajustarse a un peso máximo específico para evitar sobrecargar los actuadores y motores encargados.

Entregables del proyecto

1. Primera Presentación del proyecto

2. Maqueta del sistema

3. Informe del proyecto (primera parte)

4. Segunda Presentación del proyecto

5. Informe del proyecto (segunda parte)

6. Redmine uta (Wiki, bitácoras y Carta Gantt)

7. Poster Promocional

8. Manual de usuario

Organización del proyecto

Personal y entidades Internas

Jefe de Proyecto: Persona encargada de supervisar y planificar las actividades del desarrollo del robot.

Documentador: Persona encargada de elaborar los documentos que registran el avance del proyecto

Programador: Persona encargada de investigar las soluciones software e implementar la codificación al robot.

Encargado de hardware: Persona que facilita el uso de los recursos y gestiona los instrumentos a utilizar.

Estructura y movimiento: Persona encargada de realizar los cálculos necesarios para el movimiento óptimo del robot.

Roles y Responsabilidades

La distribución de roles está organizado de la siguiente manera:

Brayan Garcia: Documentador, estructura y movimiento.

Pablo Varas: Jefe de proyecto, programador y encargado de hardware.

Gabriel Saldias: Estructura y movimiento.

Karen Correa: Programador.

Mecanismos de comunicación

Los mecanismos de comunicación utilizados en este proyecto son:

Correo electrónico: Correo institucional (@alumnos.uta.cl)

Cuentas en redes sociales: Grupo Whatsapp, Discord.

Repositorio: Redmine, Google drive, GitHub.

Planificación de los procesos de gestión

Planificación inicial del proyecto

● Planificación de estimaciones

Producto	Cantidad	Costo por Unidad	Costo Total
Notebook	4	$400.000	$1.600.000
RaspBerry pi 5	1	$180.000	$180.000
Modulo sensor de giroscopio	1	$5.000	$5.000
Camera Module 3	1	$70.000	$70.000
Micro SD	1	$5.000	$5.000
Sensor Ultrasónico	1	$3.000	$3.000
Servomotor HS-311	4	$13.000	$52.000
Micro servo motor SG90	2	$2.000	$4.000
Motor servo de alto torque	4	$3.000	$12.000
Impresora 3D	1	$300.000	$300.000
Protoboard 400	1	$2.000	$2.000
Power bank	1	$9.000	$9.000
Total		$2.232.000	

Tabla n°1: La tabla describe su cantidad comprada y el costo de cada producto.

Recursos software gratuitos

- Python
- ROS
- Ubuntu
- VsCode

● Planificación de Recursos Humanos

Encargado	Cantidad por Rol	Número de horas por semana	Sueldo por hora	Sueldo por mes
Jefe de proyecto	1	20	$12.500	$1.000.000
Programador	2	20	$11.250	$900.000
Documentador	1	10	$8.000	$320.000
Estructura y movimiento	2	20	$11.000	$880.000
Encargado de Hardware	1	5	$10.000	$200.000
Total por mes:			$3.300.000	

Tabla n°2: La tabla muestra los roles involucrados en el proyecto y su respectivo sueldo de acuerdo a las horas de trabajo en el proyecto.

Listas de Actividades (Carta Gantt)

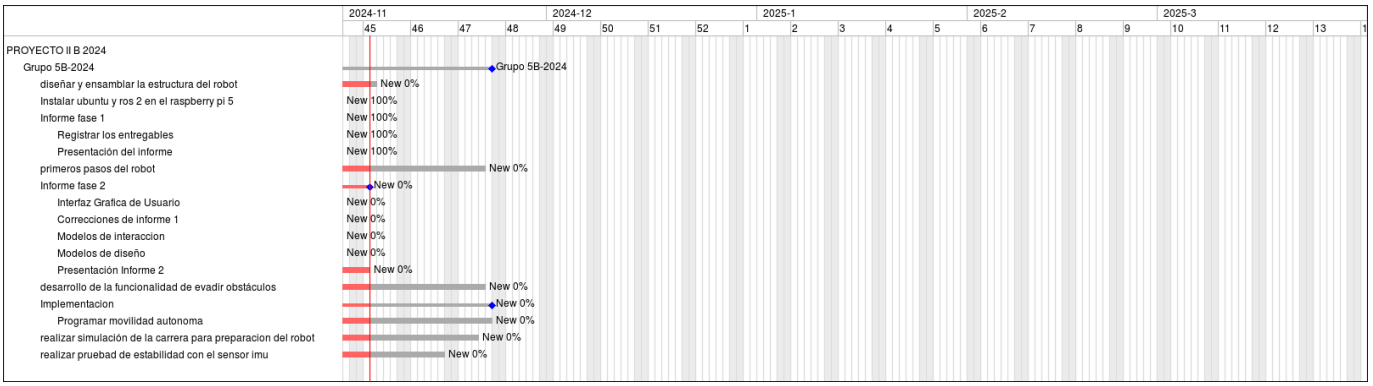

fig n°2: Lista de actividades con sus respectivas fechas a cumplir.

Planificación de la gestión de riesgos

Se valora el impacto de cada riesgo y se establece una categoría. Dichas categorías son:

1. CATASTRÓFICO

2. CRÍTICO

3. MARGINAL

4. DESPRECIABLE

Riesgos	Probabilidad de Ocurrencia	Nivel de Impacto	Acción Remedial
Falta de equipo y disponibilidad del material	60%	2	Comprar el equipo necesario para la implementación del robot.
Falta de Presupuesto	50%	2	Adaptar el diseño con los materiales proporcionados por el equipo
Falta de conocimiento del personal.	40%	2	Organizar reuniones de capacitación sobre las tecnologías que se abordan en el proyecto.
Falla de algún componente del robot	40%	2	Comprar un componente nuevo que reemplace al componente fallido.
Abandono de un miembro del equipo	30%	2	Reorganizar las actividades con los integrantes restantes.
No cumplir con las tareas en las fechas planificadas.	70%	3	Reorganizar las fechas de acuerdo al avance e información que se entrega del proyecto.
Problemas con la impresión 3D de piezas	50%	3	Asegurar la calidad del filamento y de la impresora 3D. Tener piezas de repuesto pre impresas y listas para usar en caso de fallos de fabricación.
Errores en la integración del hardware y software	50%	3	Llevar a cabo pruebas de integración continuas y tener un equipo dedicado a la solución de problemas de compatibilidad. Documentar exhaustivamente las interfaces entre hardware y software.

Tabla n°3: La tabla describe los posibles factores de riesgos que pueden ocurrir en el proyecto y la acción remedial para dicho riesgo.

Planificación de los procesos técnicos

Modelos de proceso

Requerimientos

Requerimientos Funcionales

Los requerimientos funcionales descritos incluyen acciones específicas que el sistema debe cumplir para el proyecto "Circuitron", un robot bípedo humanoide:

- Mover Robot: El robot debe responder al movimiento de acuerdo con los datos recopilados e instrucciones de la Raspberry Pi.
- Estabilizar Robot: Asegura la estabilidad del robot utilizando información de un sensor giroscópico.
- Detectar Obstáculos: Permite al robot identificar obstáculos mediante un sensor ultrasónico y almacenarlos en el sistema.
- Conectar al Robot: Activa el robot y prepara el sistema para otras funciones.

Requerimientos no Funcionales

Los requerimientos no funcionales se enfocan en aspectos de desempeño y limitaciones del sistema:

- Estabilidad del Sistema: Debe mantener la estabilidad del robot en tiempo real mediante la retroalimentación de sensores y algoritmos de control.
- Optimización Continua: Se requiere que el sistema se adapte a diferentes entornos sin modificar el hardware mediante simulaciones y ajustes continuos.
- Restricciones de Energía: La autonomía del robot está limitada por la capacidad de las baterías, afectando la duración de las pruebas.
- Capacidad de Adaptación: El sistema debe funcionar eficientemente en terrenos controlados y predefinidos.
- Compatibilidad con sensores: El sistema debe integrar sensores avanzados y modernos para procesamiento en tiempo real.

Casos de uso

Diagrama de caso de uso base

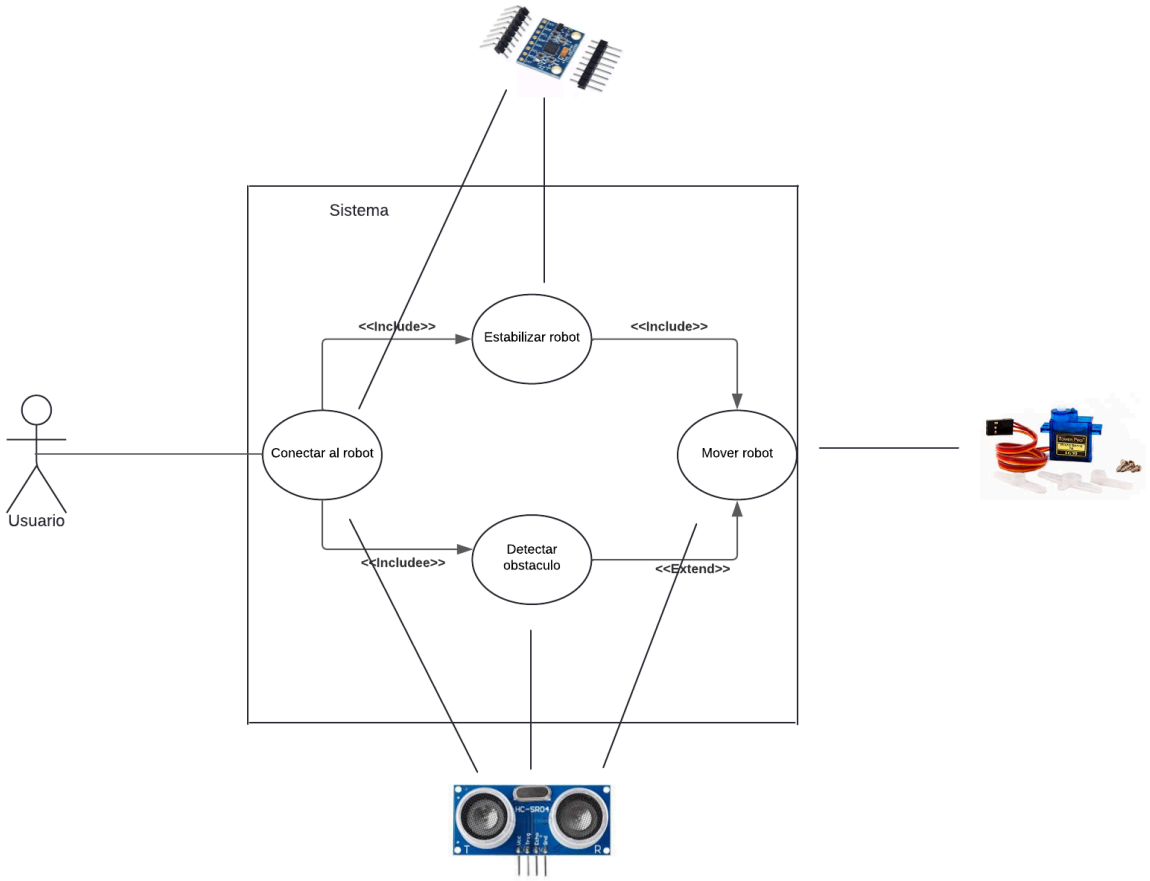

fig n°3: Diagrama que modela los agentes que actúan en el casos de uso.

Descripción de casos de uso

Nombre CUS: Conectar al robot	
Descripción:El robot prepara su sistema para realizar sus funciones.	
Actor:Usuario, sensor giroscópico, sensor ultrasónico.	
Precondición: N/A	
Flujo Principal: Usuario 2.- Selecciona “Conectar al robot”.	Flujo Principal: Sistema 1.- Muestra: “Conectar al robot” 3.-Muestra: ● Datos del sensor ultrasónico. ● Datos del sensor giroscópico.
Flujo Alternativo:	
Postcondiciones: El sistema da inicio a las funciones de los sensores y lo muestra.	

tabla n°4: Tabla que describe el CUS de conectar al robot con su descripción, precondición, actor y su flujo.

Nombre CUS: Detectar obstáculo	
Descripción:Este CUS permite detectar los obstáculos al robot.	
Actor: sensor ultrasónico.	
Precondición: El robot debe haber sido encendido.	
Flujo Principal: Robot	Flujo Principal: Sistema 1.- Include <<Conectar al robot>> 2.- El sensor ultrasónico obtiene los datos de distancia. 3.-El sistema recepciona y almacena el dato. 4.- Muestra la distancia a la que se encuentra el obstáculo.
Postcondiciones: El sistema tiene los datos para evitar el obstáculo.	

tabla n°5: Tabla que describe el CUS de detectar obstáculo con su descripción, precondición, actor y su flujo.

Nombre CUS: Estabilizar Robot	
Descripción:Este CUS permite que el robot permanezca estable a través de la información recibida del sensor giroscópico.	
Actor: sensor giroscópico, servomotor.	
Precondición: El robot debe haber sido encendido, el sensor giroscópico y los servomotores deben estar funcionando.	
Flujo Principal: Robot	Flujo Principal: Sistema 1.- Include <<Conectar al Robot>> 2.- El sensor giroscópico obtiene los datos para estabilizar el robot. 3.- El sistema ordena a los servomotores a funcionar de acuerdo al dato recepcionado.
Flujo Alternativo:	
Postcondiciones: El robot está estable para poder moverse.	

tabla n°6: Tabla que describe el CUS de estabilizar robot con su descripción, precondición, actor y su flujo.

Nombre CUS: Mover Robot	
Descripción: El robot responde con movimiento de acuerdo a la recopilación de datos y a las instrucciones de la raspberry	
Actor: Servo Motor,Raspberry	
Precondición: Los datos para manejar la estabilidad del robot deben estar en el sistema, la información de los obstáculos debe estar en el sistema.	
Flujo Principal: Usuario 3.- Pulsa el botón mover robot. 6.-pulsa “Detener robot”.	Flujo Principal: Sistema 1.- Include<<Estabilizar Robot>> 2.- Muestra “Mover robot” y “Detener robot”. 4.- El sistema debe dar la orden de movilizarse según los datos obtenidos. 7.- El sistema detiene al robot.
Flujo Alternativo: robot encuentra un obstáculo.	4.1-extend<<Detectar Obstáculo>> 5.1-El sistema debe dar la orden de esquivar obstáculos.
Postcondiciones: Robot en movimiento con sus servomotores funcionando	

tabla n°7: Tabla que describe el CUS de mover robot con su descripción, precondición, actor y su flujo.

Diagramas de secuencia

“Conectar al Robot”

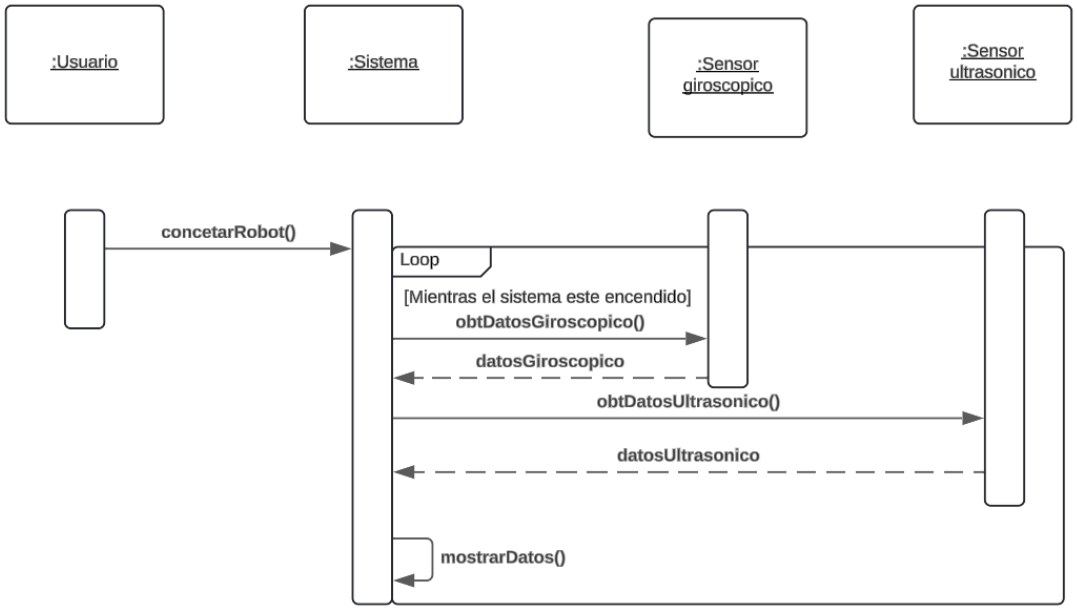

fig n°4: diagrama de secuencia que modela el CUS de conectar al robot.

“Detectar Obstáculo"

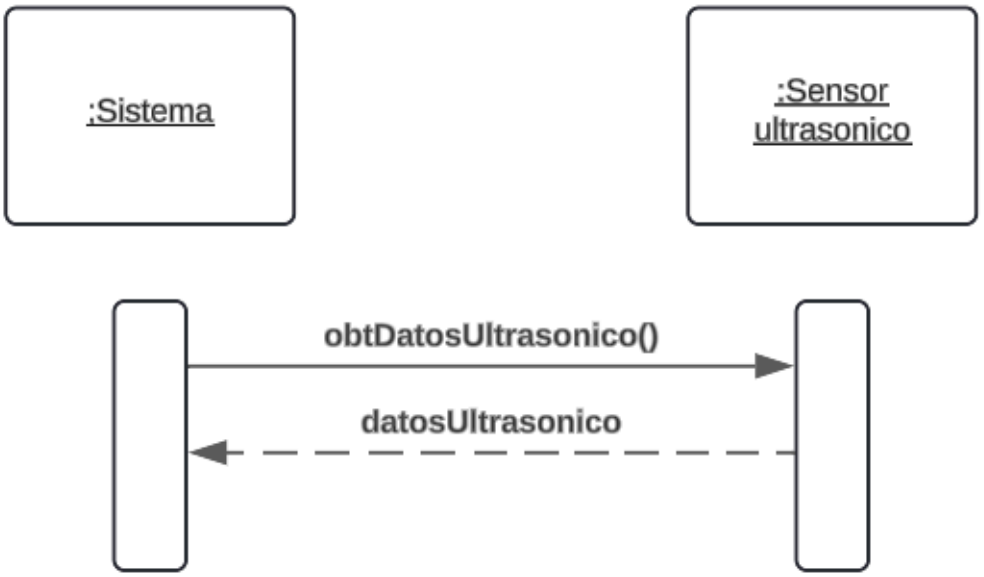

fig n°5: Diagrama de secuencia que modela el CUS detectar obstáculo.

“Estabilizar Robot”

fig n°6: Diagrama de secuencia que modela el CUS estabilizar robot.

“Mover Robot”

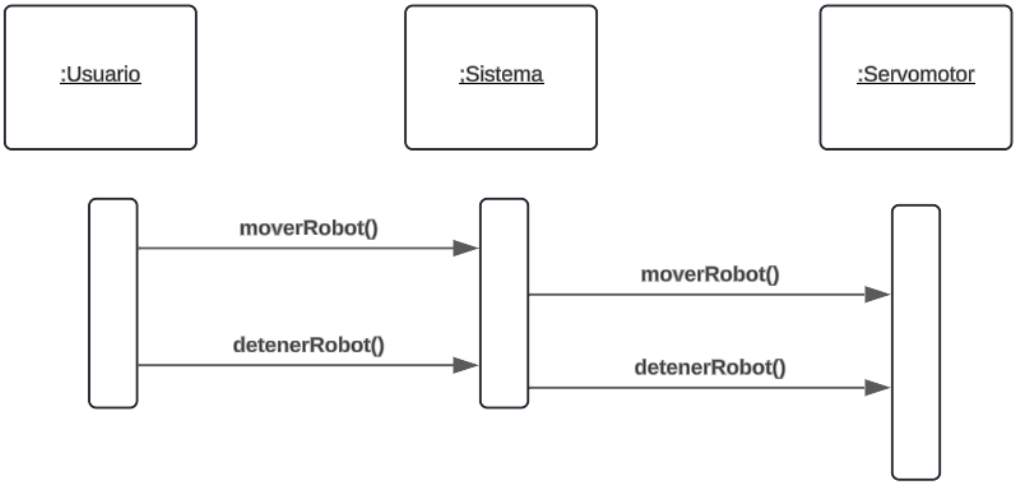

fig n°7: Diagrama de secuencia que modela el CUS mover robot.

“Mover Robot escenario en el cual se detecta un obstáculo”

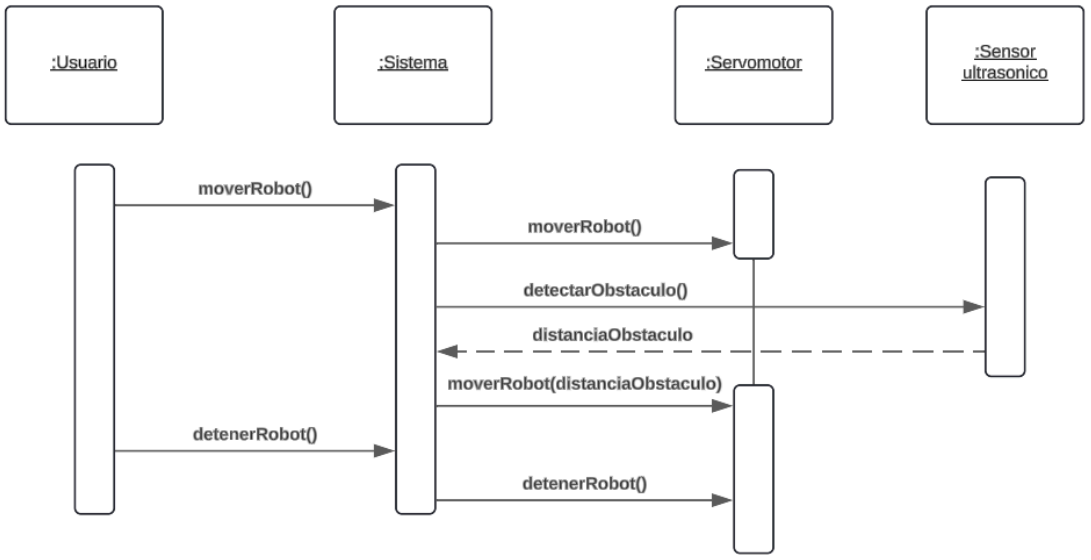

fig n°8: Diagrama de secuencia que modela el CUS mover robot en el escenario de haber un obstáculo.

Boceto Interfaz de Usuario

Paleta de Colores

El diseño de la paleta no solo se centra en la estética, sino que también tiene en cuenta la funcionalidad y la percepción del usuario. Los colores seleccionados no solo embellecen la aplicación, sino que también guían al usuario y destacan los elementos clave de interacción, mejorando así la usabilidad general de "Circuitron".

fig n°8: Boceto de la interfaz para el robot.

Bocetos de pantallas principales

El diseño simplificado de "Circuitron" no solo facilita la interacción rápida y sin complicaciones, sino que también reduce la carga cognitiva del usuario, permitiéndole enfocarse en las funciones más importantes de la aplicación.

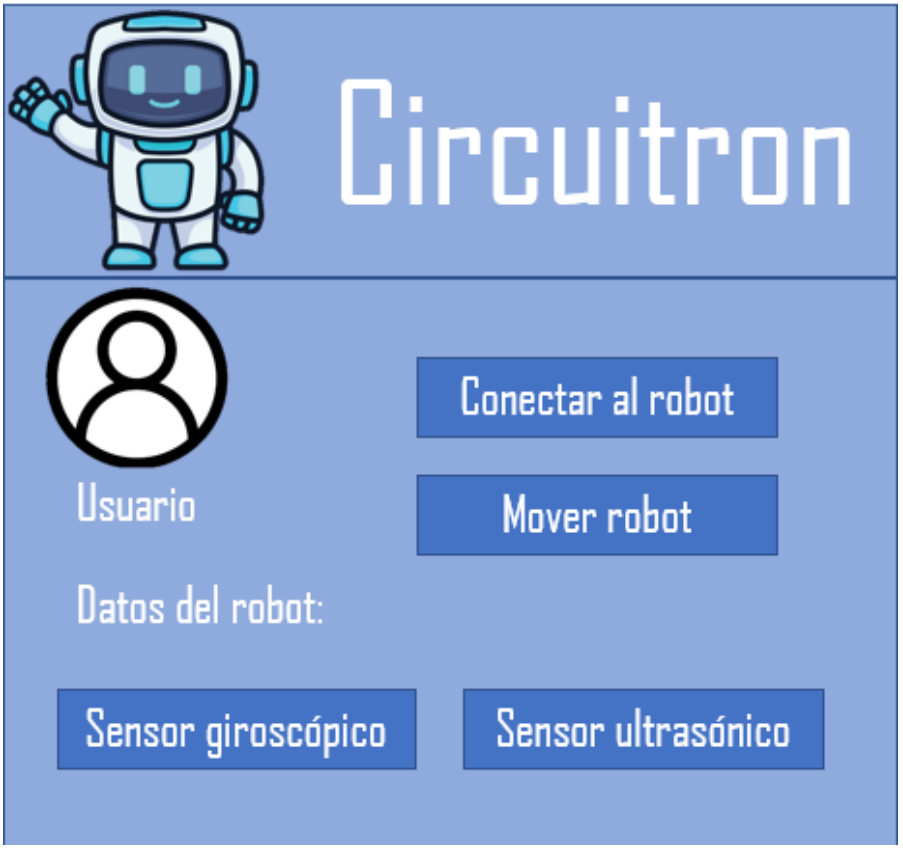

fig n°9: Interfaz gráfica para el monitoreo del robot.

Herramientas y técnicas

Herramientas:

- **Redmine**: Esta plataforma de gestión de proyectos permite planificar, supervisar y controlar las diferentes etapas del desarrollo. Facilita el seguimiento de tareas, los miembros del equipo.

- **Canva**: Herramienta de diseño gráfico y presentación que se emplea para crear un informe visualmente atractivo y presentaciones de calidad profesional.

- **Ubuntu OS**: Un sistema operativo de código abierto utilizado en la Raspberry Pi, que proporciona un entorno estable y flexible para el desarrollo y prueba de software. Su compatibilidad con herramientas de desarrollo y librerías específicas lo hace ideal para proyectos de robótica.

- **Python 3**: Lenguaje de programación seleccionado para la implementación del control del robot. Permite una programación eficiente y la integración de diversas librerías, como la GPIO, necesaria para la interacción con los componentes de hardware.

- **Thonny**: Editor de código utilizado para programar en Python. Es una herramienta intuitiva.

- **Fusion 360**: Software de diseño asistido por computadora (CAD) empleado para la creación del modelo 3D del robot bípedo.

- **RealVNC: s**oftware que permite el acceso remoto a la Raspberry Pi.asignación de responsabilidades y la visualización del progreso mediante diagramas de Gantt y otros reportes.
- **Documentos de Google**: Utilizado para la creación y colaboración en documentos en línea. Esta herramienta permite compartir y trabajar en tiempo real con otros

Técnicas:

- **Dividir para conquistar**: Descomposición de tareas complejas en subtareas manejables para optimizar el desarrollo.
- **Prototipado iterativo**: Desarrollo de versiones preliminares del robot para ajustes y mejoras.
- **Análisis de fallos**: Evaluación de posibles puntos de fallo para incrementar la confiabilidad del robot.

Implementación

1. **Sensor ultrasónic**o: El código implementa un programa en Python que permite medir la distancia entre un sensor de ultrasonido HC-SR04 y un objeto cercano utilizando una Raspberry Pi. La información obtenida es esencial para tareas de navegación, detección de obstáculos y análisis del entorno en sistemas robóticos.

```python
import RPi.GPIO as GPIO
import time


TRIG = 23
ECHO = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

def medir_distancia():
    GPIO.output(TRIG, False)
    time.sleep(0.2)


    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)


    while GPIO.input(ECHO) == 0
        inicio_pulso = time.time()

    while GPIO.input(ECHO) == 1:
        fin_pulso = time.time()

    duracion_pulso = fin_pulso - inicio_pulso


    distancia = duracion_pulso * 17150
    return round(distancia, 2)

try:
    while True:
        distancia = medir_distancia()
        print(f"Distancia: {distancia} cm")
        time.sleep(1)  # Leer cada segundo
except KeyboardInterrupt:
    print("Medición detenida por el usuario")
    GPIO.cleanup()  # Limpiar configuración GPIO al salir
```

fig n°10 Ilustra el código del sensor ultrasónico.

2. **Sensor IMU**: El sensor mide la aceleración en tres ejes (X, Y, Z) y la velocidad angular en los mismos ejes, lo que permite calcular la inclinación y analizar la estabilidad.

```python
from mpu6050 import mpu6050
import time
import math


sensor = mpu6050(0x68)

def calcular_inclinacion(aceleracion):
    """
    Calcula la inclinación del robot basándose en la aceleración en los ejes X e Y.
    Retorna el ángulo de inclinación en grados.
    """
    ax = aceleracion['x']
    ay = aceleracion['y']
    az = aceleracion['z']


    inclinacion_x = math.atan2(ay, az) * (180 / math.pi)
    inclinacion_y = math.atan2(ax, az) * (180 / math.pi)

    return inclinacion_x, inclinacion_y

try:
    while True:
        # Leer datos de aceleración del sensor
        aceleracion = sensor.get_accel_data()

        # Calcular la inclinación
        inclinacion_x, inclinacion_y = calcular_inclinacion(aceleracion)

        # Mostrar resultados
        print(f"Inclinación X: {inclinacion_x:.2f}°")
        print(f"Inclinación Y: {inclinacion_y:.2f}°")


        if abs(inclinacion_x) > 10 or abs(inclinacion_y) > 10:
            print("¡Advertencia: Robot inestable!")
        else:
            print("Robot estable.")

        time.sleep(0.5)

except KeyboardInterrupt:
    print("Programa interrumpido por el usuario.")
```

fig n°11 Ilustra el código del sensor IMU

3. **Servomotores**: Con el controlador **PCA9685** controlamos múltiples servomotores debido a su capacidad para generar señales PWM con precisión. En este caso, controlaremos 6 servomotores, simulando el movimiento de dos piernas robóticas.

```python
import time
from adafruit_pca9685 import PCA9685
from board import SCL, SDA
import busio


i2c = busio.I2C(SCL, SDA)

pwm = PCA9685(i2c)
pwm.frequency = 50

# Función para mover un servomotor
def mover_servo(channel, angulo):


    pwm_min = 150  # Valor PWM para 0 grados
    pwm_max = 600  # Valor PWM para 180 grados
    duty_cycle = int(pwm_min + (angulo / 180) * (pwm_max - pwm_min))
    pwm.channels[channel].duty_cycle = duty_cycle

# Secuencia de movimiento para simular pasos
def mover_piernas():


    pierna_izquierda = [0, 1, 2]  # Servos de la pierna izquierda
    pierna_derecha = [3, 4, 5]    # Servos de la pierna derecha


    pasos = [
        # Fase 1: Pierna izquierda hacia adelante, pierna derecha estable
        ([90, 120, 60], [90, 90, 90]),
        # Fase 2: Ambas piernas en posición media
        ([90, 90, 90], [90, 90, 90]),
        # Fase 3: Pierna derecha hacia adelante, pierna izquierda estable
        ([90, 90, 90], [90, 120, 60]),
        # Fase 4: Ambas piernas en posición inicial
        ([90, 90, 90], [90, 90, 90])
    ]

    # Ejecutar la secuencia
    for paso in pasos:
        angulos_izquierda, angulos_derecha = paso

        # Mover pierna izquierda
        for i, angulo in enumerate(angulos_izquierda):
            mover_servo(pierna_izquierda[i], angulo)

        # Mover pierna derecha
        for i, angulo in enumerate(angulos_derecha):
            mover_servo(pierna_derecha[i], angulo)

        # Pausa para observar el movimiento
        time.sleep(1)

try:
    print("Iniciando secuencia de pasos...")
    while True:
        mover_piernas()
except KeyboardInterrupt:
    print("Finalizando el programa...")
finally:
    pwm.deinit()  # Apagar el controlador PCA9685
```

fig n°12 Ilustra el código servomotores

4. **Interfaz Tkinter:**

Para la interfaz utilizamos la librería tkinter, este es el prototipo que se utilizara para la aplicación.

```
import tkinter as tk

# Crear la ventana principal
ventana = tk.Tk()
ventana.title("Circuitron")
ventana.geometry("400x450")
ventana.resizable(False, False)  # Ahora correctamente especificados ambos parámetros
ventana.configure(bg="#A5C8E1")

# Estilo para los botones
boton_estilo = {"bg": "#4A90E2", "fg": "white", "font": ("Arial", 12), "width": 15}

# Robot gráfico
robot_label = tk.Label(ventana, text="🤖", font=("Arial", 50), bg="#A5C8E1")
robot_label.place(x=150, y=10)

# Título
titulo = tk.Label(ventana, text="Circuitron", font=("Arial", 24, "bold"), bg="#A5C8E1")
titulo.place(x=120, y=100)

# Ícono de usuario
usuario_label = tk.Label(ventana, text="👤", font=("Arial", 40), bg="#A5C8E1")
usuario_label.place(x=30, y=160)

# Texto del usuario
usuario_texto = tk.Label(ventana, text="Usuario", font=("Arial", 14), bg="#A5C8E1")
usuario_texto.place(x=100, y=180)

# Datos del robot
datos_label = tk.Label(ventana, text="Datos del robot:", font=("Arial", 14), bg="#A5C8E1")
datos_label.place(x=30, y=230)

# Botones para los sensores
boton_giroscopio = tk.Button(ventana, text="Sensor giroscópico", **boton_estilo)
boton_giroscopio.place(x=30, y=270)

boton_ultrasonico = tk.Button(ventana, text="Sensor ultrasónico", **boton_estilo)
boton_ultrasonico.place(x=30, y=320)

# Botones de acción
boton_conectar = tk.Button(ventana, text="Conectar al robot", **boton_estilo)
boton_conectar.place(x=200, y=270)

boton_mover = tk.Button(ventana, text="Mover robot", **boton_estilo)
boton_mover.place(x=200, y=320)

# Iniciar la aplicación
ventana.mainloop()
```

fig n°13 Interfaz base de la aplicación

Revisión de requerimientos

Requerimientos	Cumplido	No Cumplido	Comentario
Conectividad entre el servidor y cliente.		X	Esta se realiza mediante la aplicación real vnc, que nos permite controlar la raspberry desde cualquier lugar.
Interfaz intuitiva		X	Existe una base pero falta adaptarla a los programas.
Registrar los datos de los sensores		X	Aún se deben realizar las pruebas, debido al corto tiempo y a la falta de miembros del grupo.
Lograr la movilidad del robot a través del uso de los servomotores.		X	Aún se deben realizar las pruebas, debido al corto tiempo y a la falta de miembros del grupo.
Ensamblar las piezas y estructura del robot		X	Aún faltan pocas piezas pero las más importantes ya están.

tabla n°8 Ilustra la revisión de los requerimientos cumplidos del proyecto.

Trabajo a futuro

En trabajos futuros, podemos rediseñar y optimizar la estructura de Circuitron para mejorar su funcionalidad. También se podrían incorporar sensores adicionales que aumenten su capacidad de estabilización y movilidad, permitiendo un desempeño más eficiente y adaptable en diversas condiciones.

Conclusión

Desafíos de Ingeniería y Adaptabilidad: El proyecto demostró que desarrollar robots bípedos humanoides requiere un control preciso y adaptabilidad en tiempo real. Mediante el uso de sensores avanzados y algoritmos predictivos, el robot fue capaz de adaptarse a entornos cambiantes, lo que valida la efectividad de esta combinación para futuros desarrollos robóticos.

Eficiencia en la Gestión de Recursos: A pesar de contar con un presupuesto y un tiempo de desarrollo limitados, el equipo logró implementar una solución funcional. Esto subraya la importancia de una planificación eficaz y una asignación de recursos estratégicamente pensada, especialmente en proyectos con restricciones significativas.

Limitaciones y Oportunidades de Mejora: Los desafíos de estabilidad, energía y limitaciones de tiempo resaltan áreas donde futuras iteraciones pueden enfocarse en optimizar el hardware y software del robot, especialmente en relación con la autonomía y resistencia en entornos más exigentes.

Bibliografía

[1] Laborum, “Salarios,” [Online]. Available: https://www.laborum.cl/salarios.

[Accessed: Nov. 4, 2024].

[2] MercadoLibre, “MercadoLibre Chile,” [Online]. Available: https://www.mercadolibre.cl. [Accessed: Nov. 4, 2024].

[3] Universidad de Tarapacá, “Redmine Pomerape,” [Online]. Available: http://pomerape.uta.cl/redmine. [Accessed: Nov. 4, 2024].

[4] Raspberry Pi, “Documentation,” [Online]. Available: https://www.raspberrypi.com/documentation. [Accessed: Nov. 4, 2024].

[5] Python Software Foundation, “Python Documentation,” [Online]. Available: https://www.python.org/doc. [Accessed: Nov. 4, 2024].

[6] PyPI, “RPi.GPIO,” [Online]. Available: https://pypi.org/project/RPi.GPIO/.

[Accessed: Nov. 4, 2024].