UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@ Computación e Informática

Plan de Proyecto

“Proyecto Ada”

Alumnos

Jeany Aravena

Tiara Canepa

Brandon Pizarro

Catalina Ramírez

Asignatura

Proyecto 1

Profesor(a)

Humberto Pizarro

Septiembre - 2024

1. Panorama General

1.1. Introducción

Este documento detalla la planificación a largo plazo del **Proyecto Ada**, que consiste en la fabricación de un robot con el kit LEGO Ev3 Mindstorms capaz de ser controlado manualmente y rescatar objetos de tamaño pequeño. La programación del robot se hará en el lenguaje de programación *Python* mediante la librería `ev3dev2`.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general define la meta del proyecto, lo que se espera una vez termine el plazo definido.

Diseñar y construir un robot funcional
capaz de movimiento controlado y rescate de objetos pequeños

1.2.2. Objetivos específicos

Los objetivos específicos definen el camino y las ramas que se tomarán para llegar a la meta. Algunos de los objetivos pueden ser modificados durante la ejecución del proyecto, pero es importante mantener cohesión entre ellos.

- Planificar el proyecto: roles, calendario y metas.
- Construir un modelo usando el kit LEGO Ev3 Mindstorms.
- Lograr la conexión inalámbrica con el robot.
- Programar las funcionalidades del robot.
- Agregar varios dispositivos para su control a distancia.
- Crear una interfaz para uso común.
- Adoptar satisfactoriamente la metodología AGIL.

1.3. Restricciones

La ejecución del proyecto tiene ciertas limitantes a considerar, por lo que la planificación deberá adecuarse a ellas. Algunas de las restricciones del proyecto son:

- La disponibilidad de las piezas está definida por el stock de la universidad.
- El lenguaje de programación del robot será *Python*.
- El tiempo límite del proyecto será de un semestre (4-5 meses).
- Solo se podrá trabajar con el robot dentro de la universidad.
- La plataforma de colaboración del equipo será Redmine y GitHub.

1.4. Entregables

La organización y registro del proyecto se realizará mediante documentos periódicos publicados en la plataforma Redmine.

- Bitácoras de avance cada semana.
- Calendario del proyecto.
- Informe de planificación del proyecto.

- Presentación de planificación del proyecto.

2. Organización del Personal

2.1. Descripción de los roles

Dentro del equipo se deben cubrir ciertos roles para la distribución de trabajos y carga del proyecto. Estos roles son:

1. **Jefe de grupo**: Se encarga de organizar las tareas del equipo.
2. **Ensamblador**: Se encarga de la construcción y diseño del robot.
3. **Programador**: Se encarga de implementar y mantener la comunicación con el robot y las funcionalidades que necesite.
4. **Diseñador**: Se encarga de diseñar las partes gráficas del proyecto, como el logotipo y las presentaciones.
5. **Documentador**: Se encarga de mantener actualizada la información del proyecto en la plataforma Redmine.

2.2. Personal asignado

1. **Jefe de grupo**: Tiara Canepa
2. **Ensamblador**: Brandon Pizarro, Catalina Ramírez
3. **Programador**: Jeany Aravena, Tiara Canepa
4. **Documentador**: Brandon Pizarro, Catalina Ramírez
5. **Diseñador**: Brandon Pizarro, Jeany Aravena

2.3. Mecanismos de comunicación

Para la comunicación entre los integrantes del equipo se utilizará la aplicación *WhatsApp* en donde se realizará la coordinación de las actividades y los horarios de cada miembro. Otras posibles opciones a considerar dada la necesidad serán *Discord* y la misma plataforma *Redmine*.

3. Planificación del Proyecto

3.1. Actividades

3.2. Calendario

La planificación del proyecto se realiza activamente mediante la repartición de actividades durante un tiempo estimado. El calendario, visto en la Figura 1, se describe en una carta gantt.

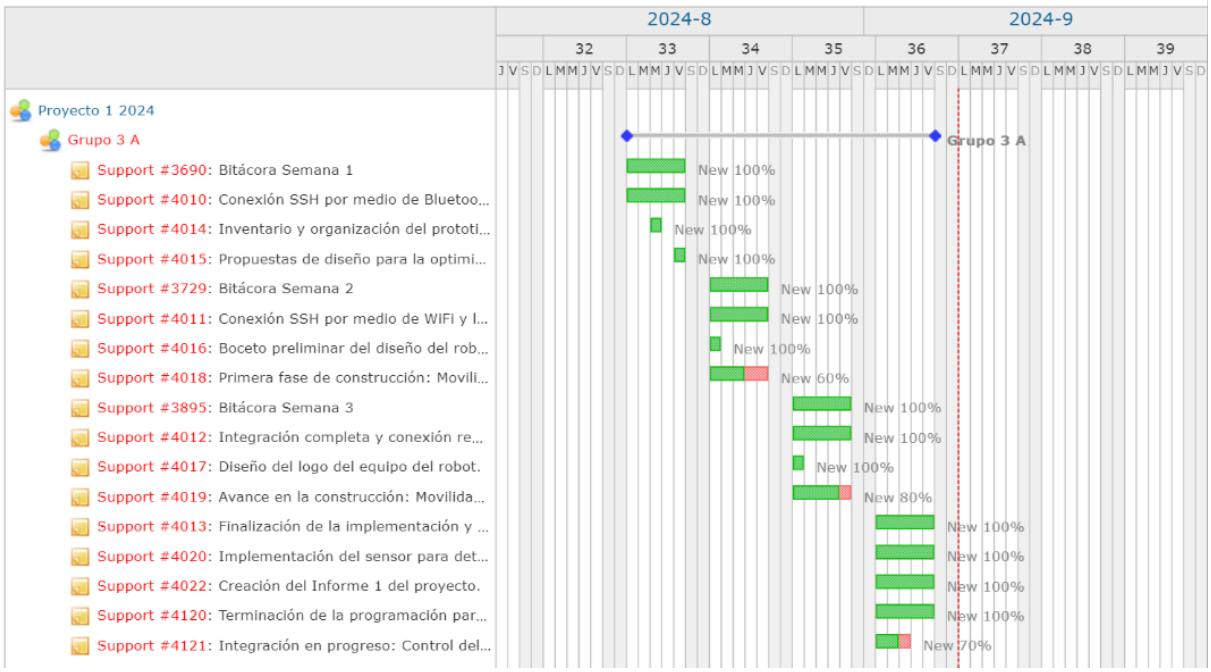

Figura 1: Calendario de actividades del proyecto

3.3. Gestión de riesgos

En la duración del proyecto pueden ocurrir inesperada e inevitablemente situaciones que pongan en peligro su avance. Para remediar estos casos se realiza un plan de acción dependiendo de su gravedad:

1. **Daño catastrófico**: El proyecto puede detenerse indefinidamente o incluso ser cancelado; las medidas a tomar deben ser eficientes y ejecutadas de forma inmediata.
2. **Daño crítico**: El proyecto puede retroceder en su avance o detenerse por cantidades considerables de tiempo; las medidas a tomar deben ejecutarse lo antes posible para evitar ramificaciones.
3. **Daño circunstancial**: El proyecto puede ser pausado o llevado por un camino no importante; las medidas a tomar tienen baja prioridad, pero deberían ejecutarse más temprano que tarde.
4. **Daño irrelevante**: El proyecto puede presentar pequeños obstáculos que no necesariamente detienen su avance; las medidas a tomar no siempre son necesarias y deberían priorizarse otras situaciones.

Riesgo	Probabilidad de Ocurrencia	Nivel de Riesgo	Medidas
Corrupción de la tarjeta SD	10%	Crítico	Formatear y reinstalar el sistema del Ev3, recurrir a respaldos de archivos
Descarga de la batería del Ev3	70%	Circunstancial	Conectar el Ev3 a una fuente de poder
Diseño incompatible con las necesidades	60%	Catastrófico	Volver a pensar en el diseño del robot y retroceder a un estado compatible
Error en el código	90%	Crítico	Arreglar los bugs del código; volver a un estado anterior si es necesario
Ausencia inesperada de algún integrante	20%	Irrelevante	Repartir la responsabilidad del miembro faltante
Piezas faltantes	60%	Circunstancial	Pedir las piezas si es que quedan en stock; en caso contrario buscar diseño alternativo
Desarme accidental considerable del robot	10%	Catastrófico	Intentar volver al diseño actual mediante material fotográfico y el manual
Cambio de horario en horas disponibles	20%	Crítico	Reagendar las actividades del horario; considerar horas extra
Fallo de los dispositivos de los integrantes	5%	Catastrófico	Recurrir a equipo prestado, contabilizar los daños; si es posible reponer dispositivos
Problemas en la conexión SSH	60%	Circunstancial	Buscar el origen del problema; si toma mucho tiempo, recurrir a ejecución directamente en el brick

4. Planificación de los Recursos

4.1. Hardware

El hardware son los dispositivos y accesorios físicos necesarios para la realización del proyecto.

- Kit LEGO Ev3 Mindstorms
- Micro SD para el almacenamiento del robot
- Notebooks para la programación y búsqueda de información
- Adaptador Wi-Fi
- Control a distancia

4.2. Software

El software son los programas y sistemas que se utilizarán para la realización del proyecto.

- Editores Visual Studio Code y Neovim
- WhatsApp
- Sistemas operativos Windows y Linux (Arch)
- Canva
- SSH

4.3. Estimación de costos

Las piezas de hardware a utilizar durante la realización del proyecto tienen los siguientes costos:

Producto	**Precio**
Kit LEGO Ev3 Mindstorms	$ 750,000
Set de Expansión LEGO Ev3	$ 250,000
Control PS3 DualShock 3	$ 30,000
Micro SD 8 GB	$ 5,000
Adaptador Wi-Fi USB	$ 10,000
Notebook Lenovo V14 G3	$ 300,000
Notebook Acer	$ 900,000
Notebook ZenBook 14x Oled	$ 600,000
Notebook Lenovo Thinkpad E14	$ 500,000
Tablet Samsung	$ 300,000
Total	$ 3,645,000

El software a utilizar durante la duración del proyecto tiene principalmente características de acceso simple; gratuito, como Visual Studio Code, o de código abierto, como Neovim, SSH o los sistemas Linux. Si bien el sistema operativo Windows requiere una licencia pagada, este costo está incluido en los productos de la Tabla 2. Por esto, no hay costo por software.

También se consideran los costos asociados con los sueldos del equipo:

Rol	Precio/Hora
Jefe de Grupo	$ 30,000
Ensamblador	$ 27,000
Documentador	$ 23,000
Programador	$ 25,000
Diseñador	$ 25,000

Naturalmente, como los miembros del equipo compartirán responsabilidad de roles los sueldos finales se ajustarán como sigue:

Integrante	Horas	Precio/Hora	Sueldo Semestral
Jeany Aravena	72	$ 28,000	$ 2,016,000
Tiara Canepa	72	$ 35,000	$ 2,520,000
Brandon Pizarro	72	$ 32,000	$ 2,304,000
Catalina Ramírez	72	$ 30,000	$ 2,160,000
Total	-	-	$ 8,890,000

En total se obtiene la siguiente estimación de costos:

Tipo	Costo Total
Hard-ware	$ 3,645,000
Sueldos	$ 8,890,000
Total	$ 12,535,000

5. Análisis y Diseño

5.1. Especificación de Requerimientos

El **Proyecto Ada** debería ser capaz de ser controlado a distancia, moverse en todas las direcciones y manipular la garra para transportar objetos pequeños. Los casos de uso, o requerimientos funcionales, que un usuario presenta al utilizar el proyecto son:

- Mover el robot en todas direcciones.
- Manejar la garra.
- Controlar al robot a distancia mediante un dispositivo electrónico.
- Hacer uso de la interfaz gráfica para mayor facilidad.

Dentro de la ejecución del proyecto, se requiere tener en consideración ciertos puntos para mantener la cohesión y el ámbito. Dentro de ellos está el diseño del proyecto y su extensión; los siguientes son requerimientos no funcionales:

- Facilitar el uso y comunicación del proyecto mediante un manual de instrucciones.
- Diseñar controles en la interfaz gráfica para el desplazamiento y manipulación de objetos.
- Mantener una conexión estable y eficiente durante su uso.
- Lograr un control inalámbrico a distancias medianas.

5.2. Arquitectura

La arquitectura del proyecto utiliza comunicaciones mediante *sockets* entre un cliente, normalmente en un computador, y un servidor dentro del robot, como muestra la Figura 2. El dispositivo procesa la entrada y envía mensajes al robot, quien actúa acorde al comportamiento que se espera.

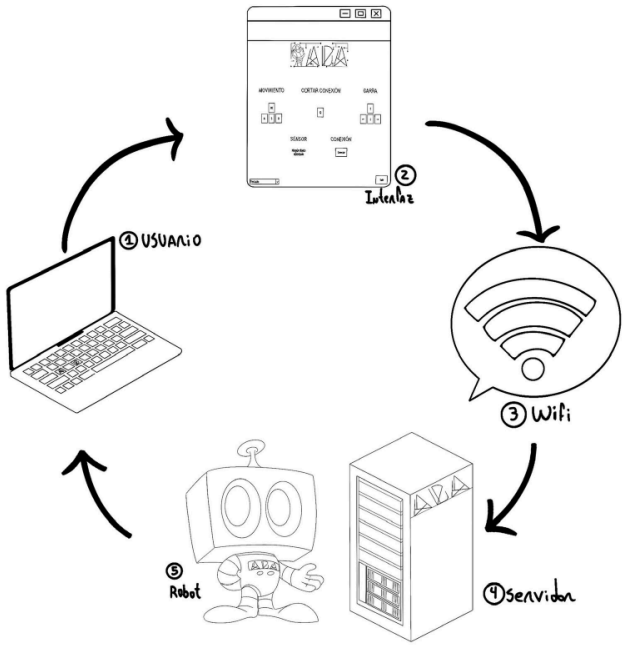

Figura 2: Arquitectura del proyecto; 1. El usuario interactúa con la interfaz, 2. La interfaz facilita la conexión a través de Wi-fi, 3. La conexión se establece con el servidor, 4. El servidor decodifica mensajes y ordena funciones al robot, 5. El robot devuelve mensajes opcionales al usuario

Para esto, el servidor posee un módulo de librería en donde se especifican las funcionalidades que puede realizar. En particular:

- Funciones de movimiento: avance, retroceso y rotación.
- Funciones de la garra: subir, bajar, abrir y cerrar.
- Funciones extra: reproducir sonidos predeterminados.

Además, para su manejo mediante dispositivos extra (controles/mandos), el cliente tiene un diseño concurrente (multi-hilo) que utiliza para la ejecución de varios archivos al mismo tiempo: uno destinado a la interfaz gráfica y otro para el uso de controles a través de la librería `pygame`.

5.3. Interfaz

La interfaz gráfica está destinada para el uso común de los usuarios. Busca facilitar el manejo del robot al no necesitar conocimiento de su implementación interna.

Al desplegarla se muestran las teclas disponibles y sus categorías (movimiento, garra y conexiones), como se ve en la Figura 3, además de un botón destinado a introducir la dirección IP del robot en caso de que la dirección guardada localmente ya no sea válida, mostrado en la Figura 4.

Figura 3: Primera pantalla de la interfaz gráfica

Figura 4: Ventana para introducir la dirección IP del robot

Una lista desplegable en la esquina inferior izquierda permite cambiar la interfaz para mostrar los botones de un control genérico, lo que se ve en la Figura 5.

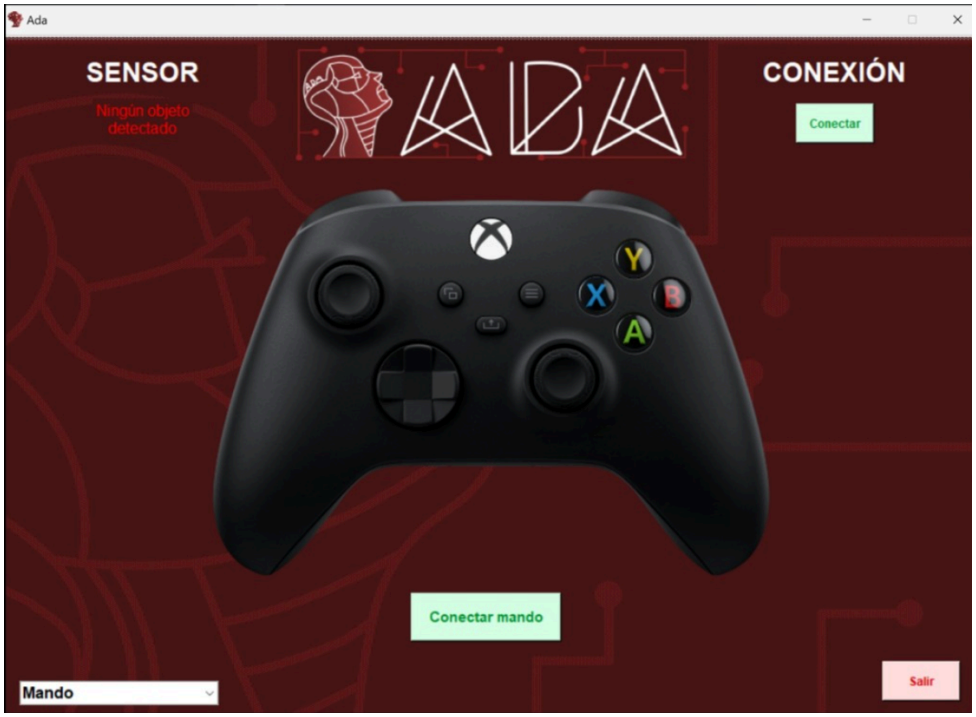

Figura 5: Segunda pantalla de la interfaz gráfica

6. Implementación

6.1. Fundamentos de Mecánica

Para el cálculo del movimiento del robot, se realizaron experimentos simples de manera de obtener su velocidad y aceleración media. Más pruebas se realizarán constantemente para controlar sus capacidades físicas.

En el experimento, partiendo desde el reposo el robot avanzó 5 metros en 10 segundos, por lo que su velocidad media está dada por $v_m = \frac{d}{t} = \frac{5}{10}\frac{m}{s} = 0.5\frac{m}{s}$. Luego, el momento en que el robot alcanza su velocidad máxima es $t = 5s$ y dada su velocidad inicial de $v_i = 0\frac{m}{s}$, se tiene que su velocidad final luego de los 5 segundos $v_f = v_m = 0.5\frac{m}{s}$. Finalmente, su aceleración media queda:

$$\begin{aligned} a_m &= \frac{\Delta x_2 - \Delta x_1}{\Delta t_2 - \Delta t_1} = \frac{0.5\frac{m}{s} - 0\frac{m}{s}}{5m - 0m} \\ &= 0.1\frac{m}{s^2} \end{aligned}$$

El transporte de objetos pequeños está modelado utilizando fuerzas de torque que debe manejar el robot en su garra. Debe ser capaz de sostener permanentemente, en particular, una pelota de ping-pong genérica.

Para esto, se asume la aceleración de gravedad $g = 9.81\frac{m}{s^2}$ y que el objeto tiene una masa $m = 0.0027$ kg. Luego, la fuerza necesaria para levantarlo será:

$$\begin{aligned} F_m &= ma = 0.0027 \text{ kg} \cdot 9.81\frac{m}{s^2} \\ &= 0.0265N \end{aligned}$$

Como la garra realiza una rotación sobre el eje del motor, levantar el objeto requiere realizar torque sobre él. Se tiene que el largo de la garra desde el eje de rotación y los dientes de la garra en donde reposa la pelota es $r = 0.2m$. Finalmente, el torque necesario que requiere realizar el robot:

$$\begin{aligned} \tau &= F_m r = 0.0265N \cdot 0.2m \\ &= 0.0053 \text{ Nm} \end{aligned}$$

Estos cálculos ignoran el roce de las partes mecánicas internas del motor y la distribución de peso en los puntos de contacto del objeto sobre la garra.

6.2. Descripción de los Programas

El proyecto consiste de los siguientes archivos:

- `server.py`
- `library.py`
- `main.py`
- `gui.py`
- `pygame_process.py`
- `client.py`

Los dos primeros archivos se encuentran dentro del robot y permiten comunicarse con el dispositivo cliente. `library.py` define sus funcionalidades —principalmente manipulando los motores—, mientras que `server.py` abre un socket por el que escucha mensajes y actúa dependiendo de la acción que se quiera realizar.

El resto de archivos constituyen el programa cliente que procesa toda entrada y la envía al servidor. El archivo de entrada `main.py` se encarga de ejecutar un proceso multi-hilo capaz de manejar los otros procesos de manera concurrente, principalmente porque el módulo `pygame` se ejecuta así nativamente.

Así, los archivos `gui.py`, `pygame_process.py` y `client.py` definen la interfaz gráfica con la librería `tkinter`, el funcionamiento con controles externos, y la comunicación con el servidor, respectivamente.

6.3. Diagramas

Se presentan aquí los diagramas que describen el diseño e implementación del proyecto.

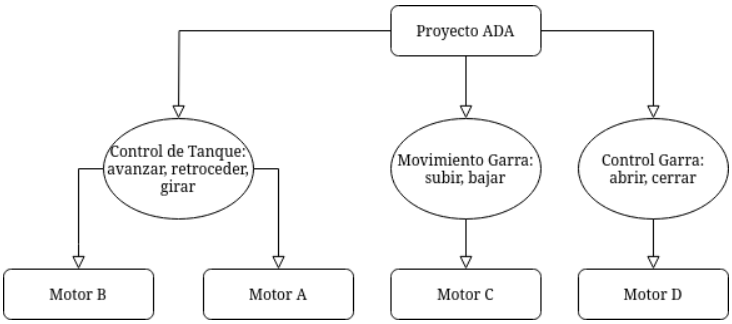

Diagrama 6: Diagrama de motores y funcionalidad

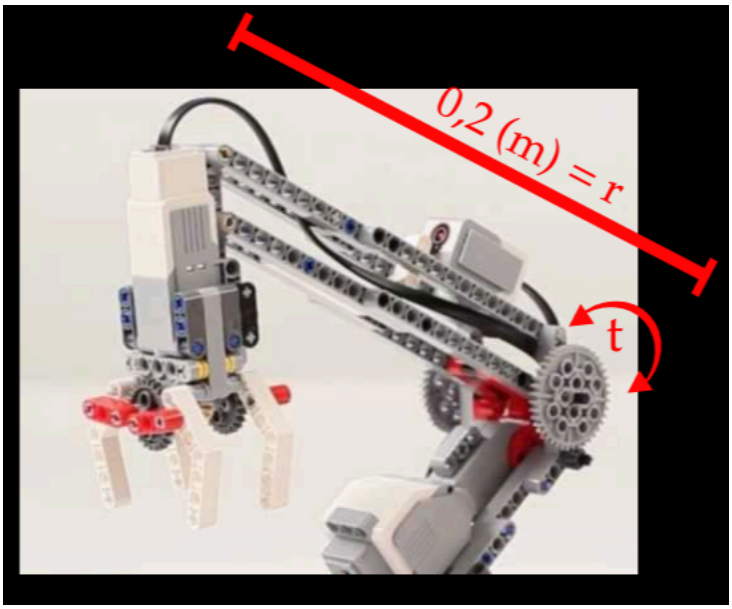

Diagrama 7: Análisis físico de la garra y el torque que produce sobre el objeto que toma

7. Resultados

7.1. Estado Actual del Proyecto

El **Proyecto Ada** actualmente es capaz de movimiento en intervalos fijos de tiempo y a velocidades constantes. Se está trabajando en una implementación de manejo mediante análogos de controles que entregará mayor versatilidad.

La interfaz gráfica está muy cerca de su versión final; solo detalles estéticos quedan por pulir, mientras que el diseño del robot ya alcanzó su última etapa, por lo que no se esperan más cambios.

El movimiento y funcionamiento de la garra ha mostrado ser satisfactorio, aunque se deben implementar límites de seguridad para la extensión de los motores.

Se ha comenzado con la realización del manual de usuario para la documentación de uso y buenas prácticas.

En general el proyecto ha tomado su curso a tiempo y no se han presentado mayores retrasos en su ejecución, así que el itinerario formulado en la planificación del proyecto ha mostrado ser fiel a la realidad y es un buen descriptor del avance.

7.2. Problemas Encontrados y Solución Propuesta

Se detallan aquí las situaciones problemáticas más importantes que se han presentado hasta ahora, y las soluciones que se propusieron:

Problemas encontrados	Soluciones propuestas
El cliente no logra la conexión con el robot	Revisar si la dirección IP del robot cambió
Durante la ejecución, el robot presenta bloqueos y no responde las instrucciones	Reiniciar el robot y el servidor para liberar los bloqueos
El programa cliente se cae si no hay un mando conectado	Refactorizar la lógica de detección de mandos con `pygame`
Un control no puede volver a ser conectado luego de desconectarse	Especificar la inicialización del control en específico dentro del código
El cliente no termina correctamente todos los procesos asociados	Buscar en la documentación de la librería multihilos de *Python* y realizar los cambios pertinentes
El cliente en Linux presenta una conexión lenta	Utilizar otro computador con Linux o depender de uno con Windows 10

8. Pruebas

Una vez la mayoría del proyecto alcanzó su etapa final se realizaron varias pruebas para determinar el correcto funcionamiento del robot, su rendimiento al cumplir tareas y los obstáculos que podrían surgir.

Cabe destacar que, en realidad, todo el desarrollo del robot se hizo en pequeños avances que se iban probando cuando era pertinente.

8.1. Descripción de las Pruebas Realizadas

- Los archivos del robot debían ser manipulables desde ambos computadores mediante asegurar las conexiones SSH. En particular, la extensión de VSCode específica para los sistemas EV3 y los programas de terminal `ssh`, `scp` y `sshfs` se utilizaron para estas pruebas.
- La conexión cliente-servidor debía funcionar para los dos computadores principales del proyecto, lo que significaba asegurar el mismo funcionamiento esencial en los sistemas Linux y Windows 10. Esto permitiría manejar el robot de forma versátil sin depender de un dispositivo específico. Para esto, los archivos de cliente debían estar actualizados en ambos computadores, que se logró con GitHub.
- La funcionalidad del robot debía ser satisfactoria. El robot sería capaz de moverse en todas direcciones y con precisión aceptable, subir y bajar la garra a control del usuario, y abrir y cerrarla.
- La garra del robot debía ser capaz de tomar objetos pequeños sin problemas de resbalamiento o pérdida de control. Para esta prueba se utilizó una pelota de ping-pong genérica de unos 2.7 gramos.
- El robot debía ser capaz de ser controlado utilizando periféricos externos como controles de consola. Todas las pruebas anteriores se repetirían con estos dispositivos para asegurar el funcionamiento completo. Se utilizo un control de PlayStation 3 y un control de Nintendo Switch, ambos con conexiones Bluetooth y por cable.

8.2. Resultados de las Pruebas

- Ambos computadores fueron capaces de modificar y crear archivos dentro del robot. En particular, el dispositivo con el sistema Windows 10, que tiene el programa VSCode como editor principal, hizo uso de la extensión para el EV3 y se logró un flujo de trabajo sencillo y eficaz; el dispositivo con Linux hizo uso de los programas de terminal para el mismo trabajo, sin pérdida de rendimiento.
- Si bien ambos computadores fueron capaces de controlar el robot con el programa cliente, el dispositivo con Linux presentaba una conexión ligeramente más lenta al servidor, demostrado por un retardo al mantener las teclas presionadas por mucho tiempo. Este problema no se presentó con el dispositivo con Windows 10, y si bien se formularon posibles soluciones se pensó como un caso no urgente.
- El robot es capaz de todas las funcionalidades pensadas, sin retraso en presionar una tecla y la respuesta del robot.
- La garra es capaz de tomar objetos pequeños de forma permanente y desplazarlos si se estima necesario.
- Ambos controles fueron capaces de manejar el robot con la misma eficiencia que con el teclado. Sin embargo, por razones a las que todavía no se encontró una respuesta, el dispositivo con Linux no era capaz de mantener una conexión por Bluetooth con los controles. Esto no retrasó el desarrollo del proyecto, pero sí resultó en un problema a considerar en el futuro.

9. Conclusión

La exitosa realización del proyecto dependerá de seguir a cabalidad las proyecciones registradas en este informe. Los plazos definidos y los roles asignados son el punto de partida de un buen producto, así que el equipo se muestra determinado a tener los mejores resultados posibles.

10. Referencias

- Página principal de ev3dev
- Repositorio de la librería ev3dev2 en Python
- Referencia y documentación de la librería ev3dev2 en Python
- Referencia y documentación de la librería pygame en Python