UNIVERSIDAD DE TARAPACÁ
FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Plan de Proyecto
“Terreneitor EV3”

Alumnos: Rafael Nakata
Tomás Carvajal
Ariel Colque
Brandon Jalanoca
Joaquín Jelves

Profesor: Humberto Urrutia
Asignatura: Proyecto I

Fecha	Versión	Descripción	Autor(es)
02/09/2024	1.0	Inicio del documento	Rafael Nakata Ariel Colque Brandon Jalanoca Joaquín Jelves
04/09/2024	1.1	Actualización del documento	Rafael Nakata
05/09/2024	1.2	Actualización del documento	Rafael Nakata Brandon Jalanoca
08/09/2024	1.3	Actualización del documento	Rafael Nakata
09/09/2024	1.4	Actualización del documento	Joaquín Jelves Rafael Nakata Brandon Jalanoca
01/10/2024	1.5	Pequeña actualización de la parte de gestión de riesgos	Brandon Jalanoca Ariel Colque
1/10/2024	1.5	Creación de una nueva sección llamada “Resultados”	Brandon Jalanoca Ariel Colque
08/11/2024	2.0	Revisión y análisis del informe anterior	Brandon Jalanoca Ariel Colque
08/11/2024	2.1	Creación del Informe II	Brandon Jalanoca Ariel Colque
11/11/2024	2.2	Avance del Informe II	Brandon Jalanoca
18/11/2024	2.3	Término del informe II	Brandon Jalanoca

Índice

1. Visión General

1.1 Introducción

A lo largo del semestre, el equipo se ha propuesto diseñar y construir un robot basado en la plataforma LEGO Mindstorms EV3, con el reto de crear un sistema robótico que no solo sea capaz de desplazarse de forma autónoma, sino también de agarrar y transportar una pelota de ping-pong utilizando una garra o un gancho. El control de este robot será gestionado a través de una interfaz desarrollada en Python, permitiendo que el usuario interactúe con él de manera eficiente.

Este informe detalla el proceso de planificación, organización y progreso del equipo, destacando la distribución de tareas, la selección de algoritmos y componentes clave, y las estrategias empleadas para cumplir con los objetivos del proyecto. Además, se incluye un análisis de las primeras impresiones del equipo y la investigación necesaria para mejorar tanto el diseño físico del robot como el desarrollo de la interfaz de control, asegurando que el robot sea capaz de realizar sus tareas de manera exitosa.

1.2 Objetivos

1.2.1. Objetivo General

Crear y programar un robot EV3 que pueda desplazarse de manera autónoma, recoger una pelota de ping-pong con una garra o gancho, y transportarla a otro lugar, controlado a través de una interfaz gráfica interactiva diseñada en Python.

1.2.2. Objetivos Específicos

- Explorar el Set de Lego Mindstorms EV3 para diseñar y construir un robot que cumpla con los requisitos de movimiento y manipulación de objetos.
- Ensamblar un robot con características de estabilidad y capacidad para moverse y realizar tareas específicas, como el agarre y traslado de objetos.
- Investigar y aplicar el uso de la librería Python junto con ev3dev para el control del robot desde una interfaz gráfica.
- Desarrollar una interfaz gráfica para el usuario utilizando la librería tkinter, facilitando el control de las funciones del robot.
- Probar y ajustar algoritmos que permitan al robot realizar tareas de manera eficiente, como la manipulación precisa de la pelota de ping-pong

1.3 Restricciones

- Se debe programar exclusivamente en Python.
- Se debe utilizar el sistema operativo Linux.
- Solo se permite el uso de la plataforma Redmine para documentos y seguimiento del proyecto.
- Obligatorio utilizar el Set de Lego Mindstorms EV3.
- Restricción de tiempo asignado al proyecto.
- El equipo está limitado a un máximo de 5 integrantes.
- Disponibilidad necesaria del robot para codificación y pruebas.
- El robot debe ser capaz de moverse hacia diferentes direcciones y debe tomar una pelota de ping-pong levantarla y dejarla en otro lugar asignado.

1.4 Entregables

- Informes del proyecto:
  - Formulación de proyecto
  - Avance de proyecto
  - Informe final

- Presentaciones de proyecto:
  - Formulación de proyecto
  - Avance de proyecto
  - Presentación final de proyecto

- Bitacoras
- Carta Gantt
- Wiki
- Manual de usuario
- Interfaz
- Robot

2. Organización del Personal

La coordinación dentro de un equipo resulta fundamental para avanzar en cualquier proyecto; por ende, es crucial distribuir las responsabilidades de manera efectiva para alcanzar con éxito los objetivos planteados.

2.1 Descripción de Roles

A continuación vamos a presentar los roles que tiene cada uno de los integrantes en el proyecto y cuál es su función.

- **Ensamblador:** Se encarga de construir el robot y ver la mejor forma para que el robot funciones de manera correcta, también se encarga de ver todas las extremidades y la mejor posición para el gancho que agarrara la pelota de ping-pong.
- **Diseñador:** Se encarga de tomar, grabar videos y mostrar los avances que se están haciendo con el robot.
- **Documentador:** Se encarga de documentar todos los avances que se están haciendo mediante informes, bitácoras y presentación.
- **Jefe de grupo:** Representante del equipo y mantener bien organizado lo que esté haciendo cada integrante de este.
- **Programador:** Se encarga de implementar los algoritmos necesarios para el funcionamiento del robot.

2.2 Personal que cumplirá los roles

Estas serán las personas encargadas de tomar el rol que se le asignó a cada miembro.

- **Jefe de grupo:** Rafael Nakata
- **Ensamblador:** Tomás Carvajal, Joaquín Jelves
- **Documentador:** Brandon Jalanoca
- **Programador:** Rafael Nakata, Ariel Colque
- **Diseñador:** Tomás Carvajal, Joaquín Jelves, Rafael Nakata

2.3 Medios de Comunicación

El medio que se están usando para comunicarse y mandar información diversa sobre el proyecto es WhatsApp, es una aplicación que se tiene a la mano y es rápido el pasar información en esta plataforma por lo que cada integrante puede enviar y comentar diferentes tipos de temas, ya sea alguna sugerencia o duda sobre el proyecto, en este medio se manda las fotos, vídeos e imágenes sobre el robot.

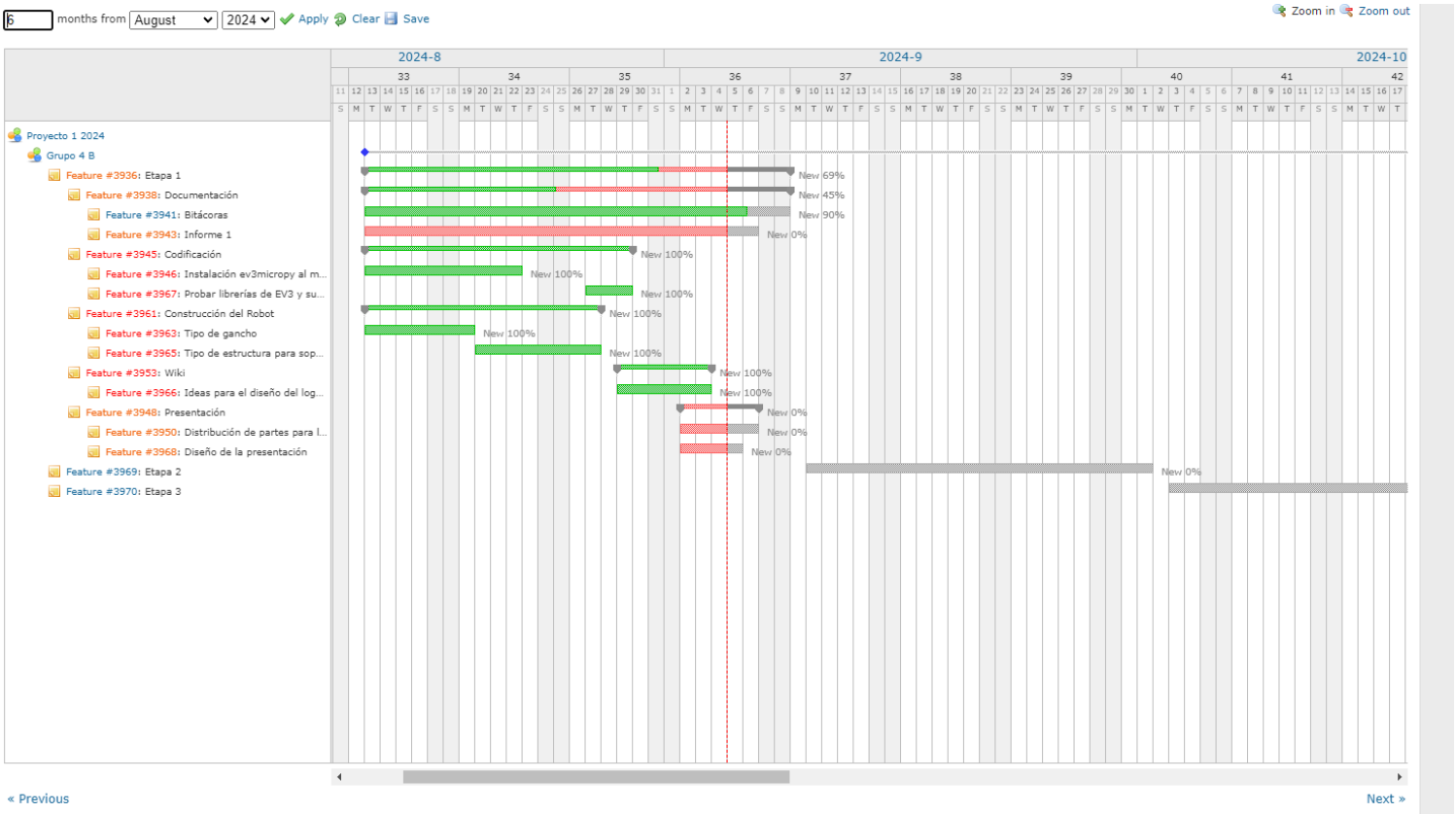
3. Planificación del Proyecto

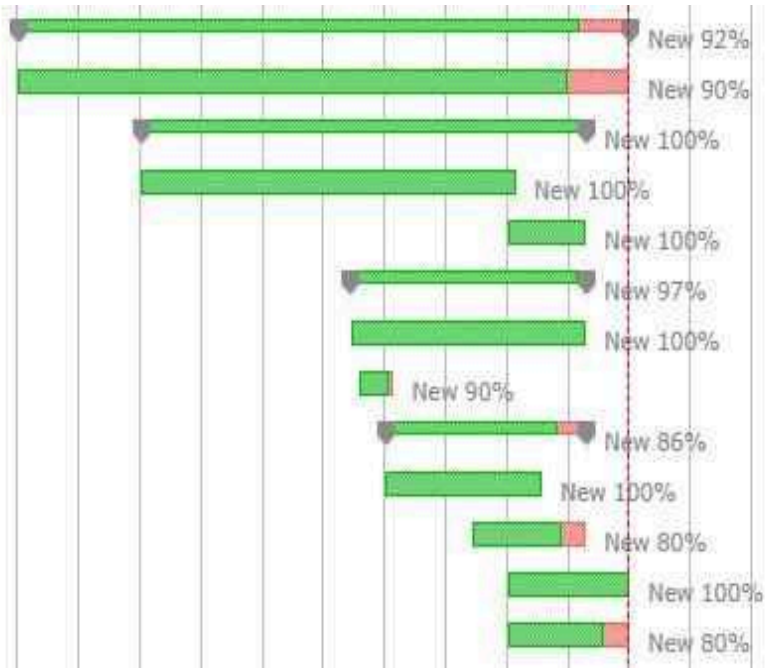
3.1 Planificación del Proyecto

Nombre	Descripción	Responsables	Producto
Formulación del proyecto.	Se definen los roles de cada miembro del grupo.	Todo el grupo.	Concretado
Avance en el primer modelo del Robot I.	Se elabora el primer modelo del Robot.	Todo el grupo.	Concretado
Redacción de las bitácoras.	Se registra lo trabajado en la semana.	Brandon Jalanoca	Concretado
Elaboración de la Carta Gantt	Planificación de las actividades a lo largo del semestre.	Rafael Nakata	Concretado
Modificación al modelo del robot.	Se actualiza el tipo de gancho del robot.	Rafael Nakata Joaquin Jelves	Concretado
Experimento con la movilización del robot	Se realiza una prueba de movilidad con el robot	Rafael Nakata Joaquin Jelves	Concretado
Videos y Fotos	Se toman fotos y videos de los avances del robot	Rafael Nakata Tomas Carvajal	Concretado
Conexión	Se establece la conexión entre el robot y el servidor	Rafael Nakata	Concretado
Elaboración del 1er informe	Se elabora el primer informe	Rafael Nakata Brandon Jalanoca	Concretado
Creación de la 1ra presentación	Se elabora la primera presentación	Rafael Nakata Ariel Colque	Concretado
Estimación de Costos	Calcular el presupuesto del proyecto	Todo el grupo	Concretado
Modificaciones a la Carta Gantt	Planificación de las actividades a lo largo del semestre	Rafael Nakata	Concretado
segunda modificación al modelo del robot	Se hacen más cambios al robot	Tomas Carvajal Joaquin Jelves	Concretado
Programación con el nuevo robot y sus movimientos	Se prueban los movimientos con la nueva versión del robot	Rafael Nakata	Concretado

Nuevas Fotos	Se toman fotos al modelo actual del robot.	Rafael Nakata Tomas Carvajal	Concretado
Comienzo de registros en la wiki	Se capturan y comparten ideas e información del proyecto.	Brandon Jalanoca	Concretado
Codificación del servidor	Encargado de la comunicación entre la interfaz y el robot.	Rafael Nataka	Concretado
Creación de la interfaz	Creación de la interfaz gráfica para los movimientos del robot	Rafael Nakata	Concretado
Conexión	Establecer la conexión con la interfaz y el robot.	Rafael Nakata	Concretado
Codificación de la interfaz	Interfaz gráfica con la que interactúa el cliente.	Rafael Nakata	Concretado
Actualización de la wiki	Se agrega más información a la wiki	Brandon Jalanoca	Concretado
tercera modificación al modelo del robot	Se hizo un cambio a la garra del robot	Tomas Carvajal Joaquin Jelves	Concretado
Codificación para el nuevo modelo y garra	Se codificó en base al nuevo modelo del robot	Rafael Nakata	Concretado
Se revisó el primer informe	Se volvió a revisar el 1er informe para comenzar el 2do	Brandon Jalanoca	Concretado
Elaboración del segundo informe	Se agregan aspectos trabajados en la fase 2	Brandon Jalanoca Ariel Colque	Concretado
Creación de la segunda presentación	Presentación que muestra los puntos más importantes en esta fase.	Ariel Colque Brandon Jalanoca	Concretado

3.2 Carta Gantt





3.3 Gestión de Riesgos

A continuación, se presenta una tabla que describe los obstáculos enfrentados en las primeras etapas del proyecto. El impacto de los riesgos se ha clasificado en cuatro niveles de gravedad:

1. **Impacto catastrófico:** Requiere una intervención inmediata, ya que puede llevar a la suspensión indefinida del proyecto.
2. **Impacto crítico:** Es necesario tomar acciones correctivas para evitar que el proyecto sufra retrasos significativos en múltiples fases.
3. **Impacto moderado:** El riesgo debe ser resuelto de inmediato, ya que podría retrasar el avance de una fase fundamental del proyecto.
4. **Impacto menor:** Se trata de un problema de poca relevancia que no requiere atención urgente y puede resolverse en cualquier momento sin afectar el progreso general.

Riesgo	Probabilidad De Ocurrencia	Nivel de Impacto	Acción Remediar
Escasez de Piezas	80%	4	Consultar al personal en el cargo.
Descarga de la Batería EV3	70%	4	Se recarga la batería con su cargador.
Enfermedad de personal	30%	4	Se reorganizan las funciones en el día para suplir al integrante enfermo.
Incumplimiento de las tareas	50%	2	Un miembro de otro rol ayuda al integrante con la tarea atrasada.
Miembros dejan el proyecto	10%	1	Se reorganizan los roles y los cargos del proyecto.
Daños al Robot por caída o algún accidente	60%	2	El ensamblador vuelve a armar la parte dañada o todo el robot mediante las fotos guardadas por el documentador.
Actualización en la arquitectura del robot por no cumplir con lo requerido en el proyecto	70%	2	Buscar un modelo más efectivo que cumpla con el objetivo del proyecto y volver a armar el robot.

Pérdida de la tarjeta micro SD	40%	3	Comprar un SD nuevo e informar al personal a cargo.
Error de código	80%	3	Investigar dónde está fallando el código y escribirlo de nuevo.
Equipo defectuoso	10%	3	Informar al profesor y solicitar materiales o equipo nuevo.
Conflicto grupal	40%	2	El jefe del grupo solicita una reunión grupal y escucha ambas partes para poder llegar a un consenso. Después de la reunión el jefe de grupo toma decisiones como cambio de roles, etc.
Retiro de ssd de la caja ev3	15%	1	Decirle a los demás integrantes del grupo que no se debe retirar la ssd.
Fallas con la conexión al servidor	30%	1	Tratas de solucionarlo lo mas rapido posible
Interrupción de estudios académicos	20%	2	Seguir con el avance en la medida de lo posible, estableciendo horarios para el desarrollo del proyecto.

4. Planificación de Recursos

En este apartado vamos a mostrarles lo que se está usando y los costos de cada instrumento tecnológico.

4.1 Hardware

Los instrumentos que se usaron en este proyecto son los siguientes:

- Tarjeta MicroSD.
- Robot EV3 Mindstorm.
- Adaptador MicroSD.
- Notebook.
- Celulares.

4.2 Software

Los software utilizados en este proyecto son lo siguientes:

- Visual Studio Code: editor de código
- WhatsApp; Medio de comunicación
- Linux: Sistema operativo
- Python: Lenguaje con el cual se trabaja
- Canva: Herramienta de diseño
- EV3 dev (ev3dev.org): Pagina donde se descargan cosas necesarias para la programación del proyecto

4.3 Estimación de costos

Costo de Hardware:

Productos	Cantidad	Precio	Categoría
Notebook Personal	2 unidades	$530.000 c/u	Hardware
Notebook de la universidad	2 unidades	$500.000 c/u	Hardware
Micro SD (8GB)	1 unidad	$5.000	Hardware
Kit Lego MINDSTORMS (EV3)	1 unidad	$1.300.000 c/u	Hardware
Total		$3.365.000	

Costo de Software:

Como no se invirtió dinero en ningún implemento de software, no se hizo una tabla ya que todo lo que usamos se usó de forma gratuita.

Costo Trabajador:

Cargo	Valor Horas Trabajadas	Horas Trabajadas	Horas extras trabajadas	Horas totales trabajadas	Sueldo mensual	Sueldo Total
PROGRAMADOR	70.000	20	8	28	$1.960.000	$9.800.000
ENSAMBLADOR	45.000	20	8	28	$1.260.000	$6.300.000
JEFE DE GRUPO	100.000	24	8	32	$3.200.000	$16.000.000
DOCUMENTADOR	45.000	15	8	23	$1.035.000	$5.175.000
DISEÑADOR	55.000	20	8	28	$1.540.000	$7.700.000
COSTO TOTAL	X	X	X	X	$8.995.000	$44.975.000

Destacado:

- **La contabilización de las horas trabajadas comienza a partir de la formación del grupo de trabajo.**
- **Para la categorización de las horas de trabajo, se tuvo en cuenta el tiempo de trabajo en clases.**
- **Para la categorización de las horas extras, se tuvo en cuenta el tiempo en las que se trabajó fuera del horario de clase, pero dentro del mismo departamento y en algunos casos en sus respectivos hogares.**

Costo total:

Costo Hardware	$3.365.000
Costo Software	$0
Costo Empleados	$44.975.000
Total	$48.340.000

5. Análisis - Diseño

5.1 Especificación de requerimientos.

5.1.1 Requerimientos funcionales

- Crear un robot que se comunique a través de wifi y permitir que el usuario lo controle a través de una interfaz gráfica basada en Python.
- La capacidad de moverse hacia adelante, atrás, izquierda y derecha.
- Es necesario que la interfaz gráfica ofrezca funciones específicas, como desplazarse, agarrar y soltar.

5.1.2 Requerimientos no funcionales

- El proyecto debe incluir un manual detallado que aporte instrucciones detalladas sobre cómo usar correctamente el robot.
- La interfaz gráfica tiene botones específicos para controlar el desplazamiento del a debe contar con robot y una sección con botones para el agarre del robot (agarrar y soltar).

5.2 Arquitectura.

Internet:

- Para que puedan comunicarse, ambos dispositivos deben estar conectados a la misma red Wi-Fi.

Servidor:

- Encargado de la conexión remota del cliente con el robot, el cual permanece en espera de alguna conexión entrante.

Cliente:

- La interfaz se conectará al servidor del robot y el usuario podrá controlarlo desde una PC.

Prototipo:

- El robot recibe la acción del usuario y la ejecuta de acuerdo con las instrucciones.

Interfaz y/o Control:

- Interfaz gráfica que el usuario usará para controlar al robot.

5.3 Interfaz


TERRENEITOR
W
A
S
D
I
I

6. Programas

Codificación de los programas que componen al robot

6.1 Descripción de programas

Servidor

Server.py

```python
#!/usr/bin/env python3
import socket
from Funciones import *
s = socket.socket()
print("Socket creado")
port = 8080
s.bind(('', port))
print("El socket se creo con puerto: {}".format(port))
s.listen(5)
print("EL socket is listening....")
connect, addr = s.accept()
print("Se conecto a  {}".format(addr))
while True:
    rawByte = connect.recv(1)
    char = rawByte.decode('utf-8')
    if (char == 'w'):
        Adelante()
    if (char == 's'):
        Atras()
    if (char == 'd'):
        Derecha()
    if (char == 'a'):
        Izquierda()
    if (char == ' '):
        Parar()
    if (char == 'q'):
        print("Terminada la sesion...")
        break
```

Funciones

Funciones.py

```python
#!/usr/bin/env python3
from ev3dev2.motor import LargeMotor, MediumMotor, OUTPUT_A, OUTPUT_B, OUTPUT_C, OUTPUT_D, SpeedPercent, MoveTank
from ev3dev2.sound import Sound

# Sonido para alertas
sound = Sound()

# Motores
ruedas = MoveTank(OUTPUT_A, OUTPUT_B)  # Motores para las ruedas izquierda y derecha
brazo = LargeMotor(OUTPUT_D)           # Motor para el brazo

# Función para mover hacia adelante
def Adelante():
    ruedas.on(SpeedPercent(100), SpeedPercent(100))  # Ambos motores a 100%

# Función para mover hacia atrás
def Atras():
    ruedas.on(SpeedPercent(-70), SpeedPercent(-70))  # Ambos motores hacia atrás a 70%

# Función para girar a la izquierda
def Izquierda():
    ruedas.on(SpeedPercent(70), SpeedPercent(-70))   # Rueda izquierda a 70% y derecha a -70% para girar

# Función para girar a la derecha
def Derecha():
    ruedas.on(SpeedPercent(-70), SpeedPercent(70))   # Rueda izquierda a -70% y derecha a 70% para girar

# Función para mover el brazo
def Brazo():
    brazo.on_for_rotations(SpeedPercent(100), -1)  # Mueve el brazo 1 rotación en dirección negativa

# Función para poner el brazo en la posición 0° (horizontal)
def Inclinacion0():
    brazo.on_for_degrees(SpeedPercent(10), 0, brake=True)  # Mueve el brazo a 0° (posicion horizontal)
```

```python
# Función para poner el brazo en la posición 45° de inclinación
def Inclinacion45():
    brazo.on_for_degrees(SpeedPercent(10), 45, brake=True)  # Mueve el brazo a 45°

# Función para poner el brazo en la posición 90° de inclinación
def Inclinacion90():
    brazo.on_for_degrees(SpeedPercent(10), 90, brake=True)  # Mueve el brazo a 90°

# Función para parar todos los motores
def Parar():
    ruedas.stop()  # Detiene los motores de las ruedas
    brazo.stop()   # Detiene el motor del brazo
```

Interfaz

Interfaz.py

```python
import tkinter as tk
from tkinter import ttk
import socket

# Variables de control para las teclas
key_states = {
    'w': False,
    'a': False,
    's': False,
    'd': False,
    'space': False,
    'shift_l': False,
    'up': False,
    'down': False
}

# Variables para la conexión
STATUS_CONNECTION = False
PORT = 8080
client = None

# Función para enviar el comando
def send_command(command):
    if STATUS_CONNECTION and client:  # Enviar si hay conexión
        try:
            client.send(bytes([ord(command)]))
            print(f"Comando '{command}' enviado al robot.")
        except socket.error as e:
            print(f"Error al enviar el comando '{command}': {e}")
            close_connection()  # Cerrar conexión si falla el envío

    else:
        print(f"No hay conexión. Comando '{command}' no enviado.")

```

```python
# Función para manejar el evento del teclado (presionado)
def key_press(event):
    key = event.keysym.lower()
    if key == 'shift_l':  # Detectar Shift izquierdo
        key = 'shift_l'
    if key in key_states:
        if not key_states[key]:  # Solo enviar si no está presionado
            key_states[key] = True  # Marcar como presionado
            send_command(key)

    # Cerrar el programa si se presiona Esc
    if key == 'escape':
        window.quit()  # Cierra la ventana principal

# Función para manejar el evento del teclado (soltado)
def key_release(event):
    key = event.keysym.lower()
    if key == 'shift_l':  # Detectar Shift izquierdo
        key = 'shift_l'
    if key in key_states:
        key_states[key] = False  # Marcar como no presionado

# Función para conectar y desconectar del EV3
def toggle_connection():
    global STATUS_CONNECTION, client
    if STATUS_CONNECTION:
        close_connection()
    else:
        IP = "172.20.10.13"  # IP del robot EV3
        try:
            client = socket.socket()
            client.connect(('172.20.10.13', PORT))
            STATUS_CONNECTION = True
            print("Conexión establecida con el robot")
            connection_button.config(text="Desconectar")  # Cambiar el texto del botón
        except socket.error as e:
            print(f"Error de conexión: {e}")
            STATUS_CONNECTION = False
            connection_button.config(text="Conectar")  # Restablecer el texto del botón
```

```python
# Función para cerrar la conexión
def close_connection():
    global STATUS_CONNECTION, client
    if client:
        client.close()
    STATUS_CONNECTION = False
    print("Conexión cerrada.")
    connection_button.config(text="Conectar")  # Cambiar el texto del botón

# Interfaz gráfica
window = tk.Tk()
window.title("Control del Robot")
window.geometry("900x500")

# Centrar la ventana en la pantalla
window.eval('tk::PlaceWindow . center')

# Cargar imágenes de los iconos desde la carpeta 'images'
icon_w = tk.PhotoImage(file="images/W.png")
icon_s = tk.PhotoImage(file="images/S.png")
icon_a = tk.PhotoImage(file="images/A.png")
icon_d = tk.PhotoImage(file="images/D.png")
icon_up = tk.PhotoImage(file="images/W.png")  # Icono para tecla Up
icon_down = tk.PhotoImage(file="images/S.png")  # Icono para tecla Down
icon_space = tk.PhotoImage(file="images/A.png")
icon_shift = tk.PhotoImage(file="images/S.png")
bg_image = tk.PhotoImage(file="images/Fondo.png")

# Crear un Label para el fondo
bg_label = tk.Label(window, image=bg_image)
bg_label.place(x=0, y=0)

# Redimensionar imágenes de los iconos
def resize_image(image, width, height):
    return image.subsample(int(image.width() / width), int(image.height() / height))

```

```
button_size = (50, 50)
icon_w_resized = resize_image(icon_w, *button_size)
icon_s_resized = resize_image(icon_s, *button_size)
icon_a_resized = resize_image(icon_a, *button_size)
icon_d_resized = resize_image(icon_d, *button_size)
icon_up_resized = resize_image(icon_up, *button_size)
icon_down_resized = resize_image(icon_down, *button_size)
icon_space_resized = resize_image(icon_space, *button_size)
icon_shift_resized = resize_image(icon_shift, *button_size)

# Botones de dirección
btn_up = tk.Button(window, image=icon_w_resized, command=lambda: send_command('w'))
btn_up.place(x=150, y=260)

btn_down = tk.Button(window, image=icon_s_resized, command=lambda: send_command('s'))
btn_down.place(x=150, y=340)

btn_left = tk.Button(window, image=icon_a_resized, command=lambda: send_command('a'))
btn_left.place(x=70, y=340)

btn_right = tk.Button(window, image=icon_d_resized, command=lambda: send_command('d'))
btn_right.place(x=230, y=340)

# Botones para Up y Down adicionales
btn_arrow_up = tk.Button(window, image=icon_up_resized, command=lambda: send_command('up'))
btn_arrow_up.place(x=300, y=200)

btn_arrow_down = tk.Button(window, image=icon_down_resized, command=lambda: send_command('down'
btn_arrow_down.place(x=300, y=400)
```

```
# Botón para parar el robot
btn_space = tk.Button(window, image=icon_space_resized, command=lambda: send_command('space'))
btn_space.place(x=400, y=250)

# Botón para Shift
btn_shift = tk.Button(window, image=icon_shift_resized, command=lambda: send_command('shift_l'))
btn_shift.place(x=400, y=300)

# Botón para conectar/desconectar
connection_button = tk.Button(window, text="Conectar", command=toggle_connection)
connection_button.place(x=400, y=100)

# Vincular las teclas al evento key_press y key_release
window.bind("<KeyPress>", key_press)
window.bind("<KeyRelease>", key_release)

# Ejecutar la ventana principal
window.mainloop()
```

7. Resultados

7.1 Estado actual del proyecto

En el tiempo que llevamos con el proyecto se presentarán avances de este mismo.

- Estado de la versión actual del robot: El Robot estaba completo en su totalidad pero se tuvo que hacer un cambio a la garra ya que no cumplía con lo que se pedía.
- Función de movimientos implementada: El robot de momento puede mover cada una de sus ruedas por sí sola.

7.2 Problemas encontrados y soluciones

Problemas	Soluciones
Falta de materiales	Comunicarse con el responsable de la distribución para obtener un mayor suministro.
Reconstrucción frecuente del robot	Llegar a una idea precisa que cumpla con todas las características y a su vez complete los objetivos establecidos.
Complicaciones con la garra y la estructura en el del robot	Reestructuración del diseño mediante ajustes en la base y fortalecimiento en los puntos débiles del robot.
Modificación frecuente de la interfaz gráfica y funciones debido a la reconstrucción del robot.	Terminar las modificaciones del robot para dar avance al desarrollo de la interfaz y funciones.

8. Conclusión

Este proyecto nos enseñó la importancia de organizar el tiempo y las tareas de cada miembro del grupo. Al tener una buena organización, el trabajo en equipo se hizo más fácil ya que cada miembro cumplió con sus responsabilidades de manera comprometida. La Carta Gantt es clave para el proyecto, ya que ayuda a organizar el tiempo y lograr un desarrollo exitoso y eficiente. y a medida que fue avanzando el tiempo comprendimos de mejor manera el cómo reaccionar ante situaciones imprevistas que se han ocasionado en el trayecto hasta ahora.