UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@ Computación e Informática

Plan de Proyecto
“EVA-π3”

Alumno(os): Bastian Cruz
Benjamin Tamarin
Antonella Butron
Sebastian Eyraud
Josue Sucso

Asignatura: Proyecto I

Profesor: Humberto Urrutia López

09 – 2024

Tabla de Contenidos

1. Panel General

1.1. Introducción

El proyecto EVA-π3 se centra en el diseño y desarrollo de un robot construido con la plataforma Lego Mindstorms EV3. Este proyecto abarca tanto la construcción física del robot como la implementación de sistemas avanzados que integran servidores y lenguajes de programación para controlar su comportamiento. A lo largo de este informe, se presenta una documentación exhaustiva que cubre desde las fases iniciales de diseño hasta la programación de las funciones principales del robot. El objetivo es recopilar y organizar de manera estructurada los datos más relevantes obtenidos durante la creación y puesta en funcionamiento del robot, proporcionando una visión integral y detallada del proceso de desarrollo.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar y programar un robot basado en la plataforma Lego Mindstorms EV3, capaz de movilizarse, identificar y recoger una pelota. Todo el proceso será gestionado y monitorizado a través de una interfaz gráfica avanzada, que permitirá una interacción y supervisión eficiente del robot. Este objetivo busca integrar tanto el diseño mecánico como la programación avanzada, garantizando que el robot cumpla con las funciones específicas de manera precisa, robusta y confiable.

1.2.2. Objetivos Específicos

1. **Investigar el Set de Lego Mindstorms EV3:** Conocer sus componentes, características y cómo construir y programar robots, incluyendo sensores, motores y herramientas de software para proyectos de robótica educativa**.**
2. **Construir y ensamblar un prototipo:** Reunir y montar los componentes necesarios para crear un modelo funcional de un robot, asegurando que todos los elementos, como motores y sensores, estén correctamente conectados

3. **Estudiar la librería Python para EV3:** Implica aprender sobre las funciones y herramientas disponibles para programar y controlar robots construidos con Lego Mindstorms EV3 utilizando el lenguaje Python.
4. **Investigar la conexión de servidor con el robot y computador:** Implica explorar los métodos y protocolos necesarios para comunicar ambos dispositivos, permitiendo el control y la programación del robot desde el computador.

1.3. Restricciones

1. El desarrollo se realizará únicamente en Python.
2. Es obligatorio utilizar Linux como sistema operativo.
3. La gestión del proyecto y la documentación se hará a través de Redmine.
4. El proyecto cuenta con tiempo y recursos limitados.
5. El equipo está compuesto por solo 5 miembros.
6. Disponibilidad restringida del robot para pruebas.

1.4. Entregables

1. Bitácoras
2. Carta Gantt
3. Informe de Formulación.
4. Manual de Usuario
5. Presentaciones

2. Organización del Personal

La estructura organizativa adoptada es un elemento crucial en el avance del proyecto. En consecuencia, se ha procedido a una meticulosa asignación de responsabilidades y tareas entre los miembros del equipo, lo cual ha establecido a cada participante como un componente esencial en la construcción del robot y en la implementación exitosa de las funcionalidades previstas. Esta distribución estratégica de roles asegura una colaboración eficiente y el logro de los objetivos establecidos.

2.1. Descripción de los Roles

Jefe de Proyecto: Lidera la supervisión y organización general, coordina reuniones y soluciones estratégicas.

Ensamblaje: Realiza el montaje del robot, colaborando con el programador para asegurar la funcionalidad.

Programador: Se encarga del desarrollo del código y la operativa del robot, en sinergia con el técnico de ensamblaje.

Documentador: Documenta el progreso, elabora informes técnicos del proyecto, junto a las presentaciones.

Diseñador: Crea la identidad visual, logotipo e interfaz, manteniendo la cohesión estética del proyecto.

2.2. Personal que Cumplirá los Roles

Rol	Responsable	Involucrados
Jefe de proyecto	Benjamin Tamarin	Benjamin Tamarin
Ensamblador	Sebastian Eyraud	Sebastian Eyraud
Diseñador	Sebastian Eyraud	Sebastian Eyraud
Programador	Benjamin Tamarin Josue Sucso	Benjamin Tamarin Bastian Cruz Josue Sucso
Documentador	Antonella Butrón Bastian Cruz	Antonella Butrón Bastian Cruz

2.3. Métodos de Comunicación

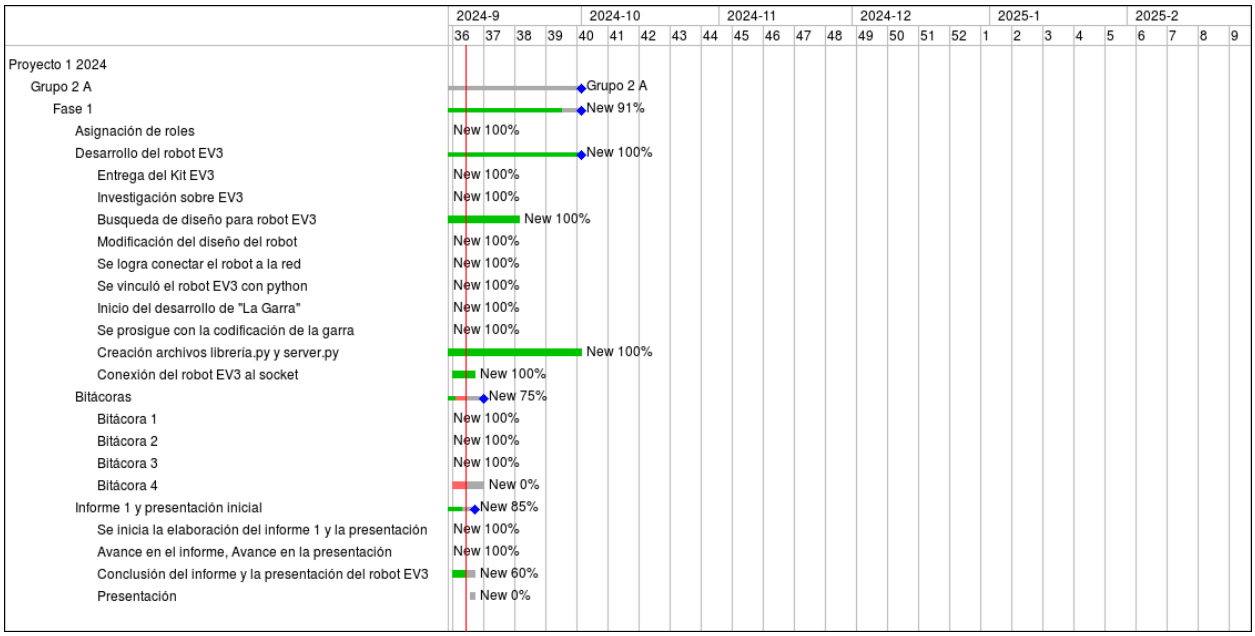
Para el desarrollo del proyecto, se emplearán WhatsApp y Discord para comunicación rápida, GitHub para gestión de código y documentación, y reuniones presenciales para discusiones detalladas, garantizando la participación activa de todos los miembros y la eficiencia del proyecto.

3. Planificación del Proyecto

3.1. Actividades

Nombre	Descripción	Responsables	Producto
Formulación del proyecto	Se analizan las piezas del Set Lego MindStorm Ev3 y se formula la primera bitácora.	Todo el equipo.	Avance del prototipo del robot y experimentación con las piezas.
Avance en el modelo del robot, funcionalidades y organización del proyecto	Se avanza con el ensamblado final del robot. Pruebas experimentales con código para el robot	Bastian Cruz Sebastian Eyraud Benjamin Tamarin	Se avanza gran parte del robot final. Se aprende acerca de la codificación y funcionamiento para el robot .
Gestión de Documentación del Proyecto	Elaboración de las bitácoras, documentación, informe y carta gantt.	Antonella Butrón Josue Sucso Bastian Cruz	Se hacen bases para la documentación y carta gantt.

3.2. Carta Gantt



3.3. Gestión de Riesgos

En esta sección se detallan los riesgos identificados durante el desarrollo del proyecto, clasificados según su nivel de impacto y probabilidad de ocurrencia. Se incluyen estrategias de mitigación y acciones correctivas específicas para cada riesgo:

- **Daño catastrófico**: Necesita intervención inmediata, ya que existe el riesgo de que el proyecto se detenga de manera indefinida.
- **Daño crítico**: Requiere acciones correctivas significativas para evitar grandes retrasos que afecten varias fases del proyecto.
- **Daño circunstancia**l: Debe ser abordado rápidamente para prevenir demoras en etapas cruciales del proyecto.
- **Daño irrelevante**: Provoca inconvenientes leves que pueden resolverse en cualquier momento sin afectar de manera significativa el progreso del proyecto.

Riesgo	Probabilidad de Ocurrencia	Nivel de Impacto	Acción Remedial
Errores en el código	50%	Daño Crítico	Revisión frecuente del código, pruebas unitarias y copias de seguridad.
Fallas de hardware	20%	Daño crítico	Componentes de repuesto y pruebas previas a la integración
Incompatibilidad de software	20%	Daño circunstancial	Verificación de compatibilidad y uso de herramientas estándar.
Retrasos en la entrega de componentes	50%	Daño crítico	Planificación anticipada y proveedores alternativos.
Interrupciones en el suministro eléctrico o acceso a laboratorios	20%	Daño circunstancial	Trabajos críticos en horas seguras y fuente de energía de respaldo.
Desmontaje accidental del robot	20%	Daño crítico	Aseguramiento del robot y pruebas en entornos controlados.
Pérdida de datos en el almacenamiento principal(Tarjeta SD)	50%	Daño crítico	Copias de seguridad regulares y dispositivos fiables.
Insuficiencia de tiempo para trabajo independiente	80%	Daño circunstancial	Gestión eficiente del tiempo, metas claras y distribución equitativa del trabajo.
La baja carga de la batería del robot	50%	Daño circunstancial	Recargar la batería completamente antes para evitar interrupciones en el funcionamiento del robot

4. Planificación de los Recursos

4.1. Hardware

Para el desarrollo del proyecto con el set Lego Mindstorms EV3, se necesitarán los siguientes recursos:

- *Set Lego Mindstorms EV3*: Un kit de robótica avanzado que proporciona los componentes necesarios para construir y programar robots personalizables y autónomos, incluyendo sensores, actuadores y la unidad de control EV3.
- *Micro SD para Lego Mindstorms*: Una tarjeta de memoria Micro SD compatible con el set de Lego Mindstorms EV3, que se utilizará para almacenar y ejecutar los programas del robot escritos en Python.
- *Computador con el sistema operativo adecuado(en este caso Linux)*: Se requieren computadoras equipadas con el software y el sistema operativo necesarios para desarrollar las instrucciones de programación para el robot.
- Computador para hacer los informes, bitácoras, carta gantt y presentaciones:Se requieren otras computadoras para asegurar que las tareas administrativas del proyecto se realicen de manera efectiva y eficiente.
- Computador para codificar en Python:Se utilizó un computador con sistema operativo Windows para indagar codificación y funcionalidades del robot, de esta manera, siendo un apoyo para el proyecto.

4.2. Software

El proyecto utilizará las siguientes herramientas de software:

- **Sistema Operativo Linux**: Proporciona un entorno de desarrollo estable y personalizable para la programación y ejecución del software del robot.
- **GitHub**: Plataforma de control de versiones y colaboración para el código fuente y la documentación del proyecto.
- **Visual Studio Code**: Editor de código fuente versátil y ligero para escribir y depurar programas.
- **Canva**: Herramienta de diseño gráfico para crear materiales visuales como logotipos e interfaces para el proyecto.

- **Microsoft Office**: Se emplea para la redacción y formateo de documentos oficiales, informes y otros materiales escritos asociados con el proyecto.

4.3. Estimación de Costos

La estimación de costos se ha realizado considerando los siguientes elementos:

Costo de Hardware:

Producto	Precio
Set Lego Mindstorm Education(EV3)	355 340
Tarjeta Memoria Micro SD	6.950
Piezas de expansión	228.285
Notebook, TECRA Z40, PROCESADOR CI5-6300U (Para el sistema operativo, Linux)	228.626
Notebook, Lenovo v14 G2 ALC (Para codificar)	340.000
Notebook, IdeaPad L340-15IRH Gaming (Para la realización de Bitácoras, Informes y Presentaciones)	830.000
Total:	1.760.575

Costo de Software:

Producto	Precio
Licencia de Canva(Equipo)	91.385/al año
Licencia Microsoft Office	10.000
Total :	101.385

Costo de Trabajador:

Rol	Horas	Horas Extra	Precio / Hora
Jefe de proyecto	72	5	65.000
Programador	72	10	45.000
Ensamblador	72	7	38.000
Diseñador	72	4	28.000
Documentador	72	3	25.000
Total	/	/	15.700.000

Total de Costo:

Costo Hardware	1.760.575
Costo Software	101.385
Costo Empleados	15.700.000
Total :	17.561.960

5. Análisis- Diseño

5.1. Especificación de requerimientos

Requerimientos funcionales:

Estos son los requerimientos que describen las funciones específicas que el sistema debe cumplir.

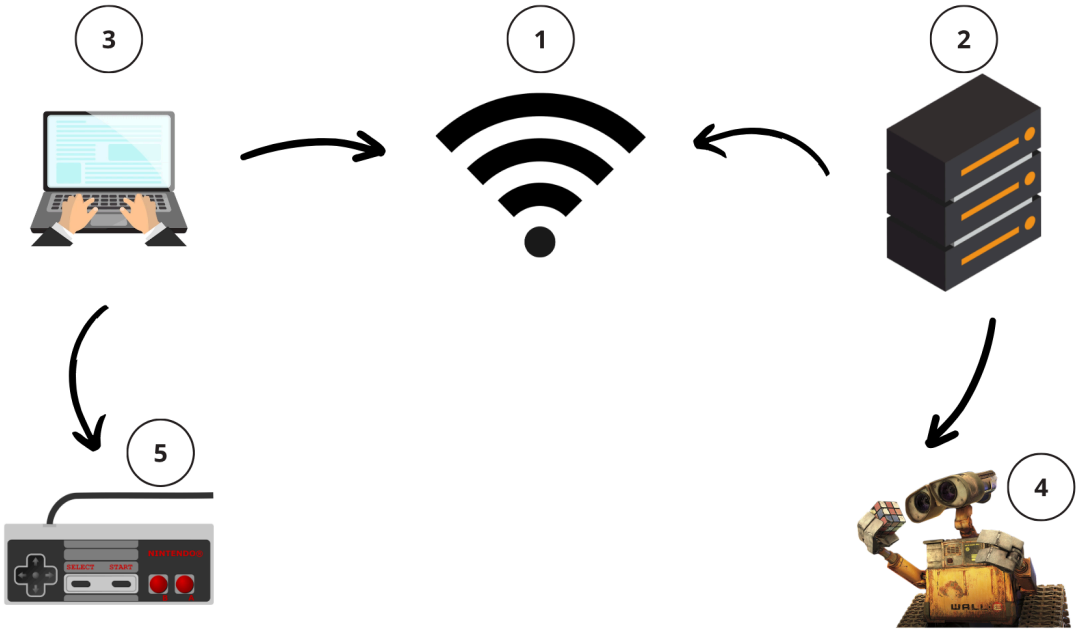
- El robot debe poder moverse en cualquier dirección.
- El robot debe ser capaz de detectar y recoger una pelota utilizando sus sensores.
- El control remoto del robot debe funcionar adecuadamente mediante una interfaz gráfica.
- El robot debe poder procesar las órdenes enviadas desde el servidor de manera eficiente y en tiempo real.
- La interfaz gráfica debe proporcionar retroalimentación visual sobre las acciones del robot.

Requerimientos no funcionales:

Aquí se definen aspectos como la calidad, rendimiento y limitaciones técnicas del proyecto:

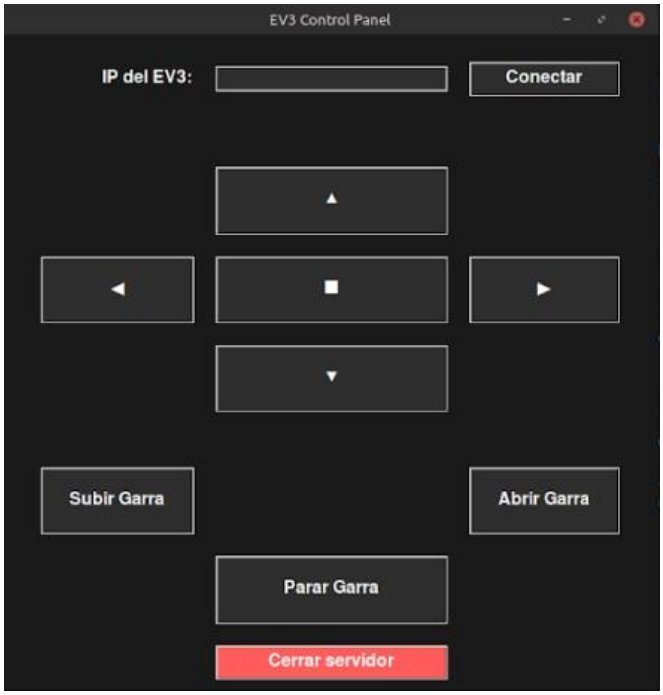
- El tiempo de respuesta entre el comando enviado y la acción del robot no debe superar los x segundos.
- El sistema debe ser compatible con Linux y con la librería Python para EV3.
- El robot debe ser resistente a fallos menores, como desconexiones intermitentes o batería baja.
- Tanto el Hardware como el software del robot deben permitir futuras expansiones o mejoras.
- El código del sistema debe estar bien documentado para facilitar actualizaciones futuras.
- El proyecto debe contar con un manual de usuario que sirva de guia para el uso del robot y su interfaz.
- La interfaz gráfica debe incluir botones específicos para el control preciso del desplazamiento y el sistema de agarre del robot, además de un sistema de desconexión integrado para mayor seguridad.

5.2. Arquitectura propuesta



1-Ambos dispositivos deben estar en la misma red Wi-Fi para que se pueda realizar la conexión.
2- Responsable de la conexión remota del cliente al robot, que permanece a la espera de una solicitud de conexión.
3- A través de una computadora, la interfaz se conectará al servidor del robot, permitiendo al usuario tomar el control.
4- El robot lleva a cabo la acción solicitada por el usuario, siguiendo las instrucciones proporcionadas.
5-Interfaz gráfica Tk que utilizara el usuario para controlar al robot.

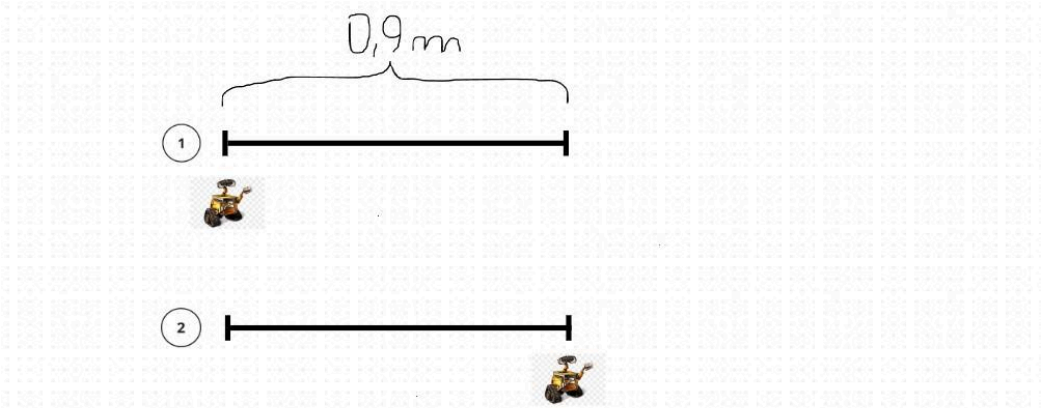
5.3. Diseño de la interfaz de usuario



6. Implementación

6.1. Fundamentos de la acciones del Robot

Velocidad de desplazamiento del robot



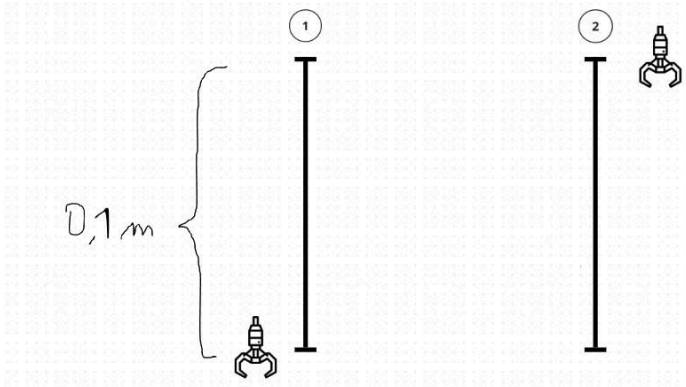
Desplazamiento del robot(d): 0.90m

Tiempo de desplazamiento(t): 10.56s

Velocidad de desplazamiento del robot (v):

$$V_R = \frac{d}{T} = \frac{0,9}{10,56} = 0,085 \frac{m}{s} \text{ o } 8,5 \frac{cm}{s}$$

Velocidad de elevación de la Garra

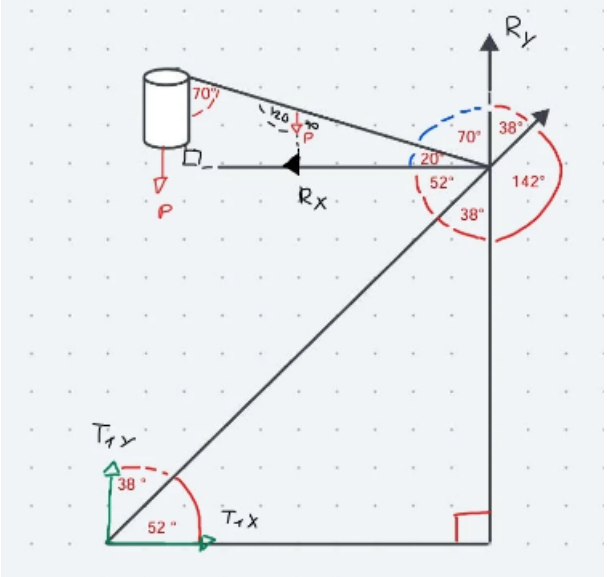


Elevación de la Garra(h): 0.1m

Tiempo de elevación(t): 2s

Velocidad de elevación de la Garra (v):

$$V_G = \frac{h}{T} = \frac{0,1}{2} = 0,05 \frac{m}{s} \text{ o } 5 \frac{cm}{s}$$



TORQUE = 0

-T1 * lARGO MOTOR * SEN(52°) + PESO BARRA * (LARGO BARRA)/2 * SEN(108°) + T2 * LARGO BARRA * SEN(120°) = 0

fx = 0

RX - T1X = 0

fy = 0

T1Y - T2 - P + RY = 0

6.2. Descripción de los programas

-EV3

```python
from ev3dev2.motor import MoveTank , OUTPUT_A, OUTPUT_B,
LargeMotor , OUTPUT_C, MediumMotor, OUTPUT_D
import time
mov_garra = LargeMotor(OUTPUT_C)
garra = MediumMotor(OUTPUT_D)
tankmoves = MoveTank(OUTPUT_A, OUTPUT_B)

def move_up():
  """Mueve el robot hacia adelante."""
  print("Avanzar")
  tankmoves.on(25, 25)

def move_down():
  """Mueve el robot hacia atrás."""
  print("Retroceder")
  tankmoves.on(-25, -25)

def move_right():
  """Gira el robot a la derecha."""
  print("Derecha")
  tankmoves.on(5, -5)

def move_left():
  """Gira el robot a la izquierda."""
  print("Izquierda")
  tankmoves.on(-5, 5)
```

```python
def handle_up():
 """Gira el robot a la izquierda."""
 print("Subiendo")
 mov_garra.on(-7)
 time.sleep(4)
 mov_garra.off()

def handle_down():
 """Gira el robot a la izquierda."""
 print("Bajando")
 mov_garra.on(7)
 time.sleep(4)
 mov_garra.off()

def handle_open():
 print("Abriendo")
 garra.on(3)

def handle_close():
 print("Cerrando")
 garra.on(-3)

def stop_handle():
 print("Deteniendo garra")
 garra.off()

def stop():
 print("Detenido"
 tankmoves.off()
```

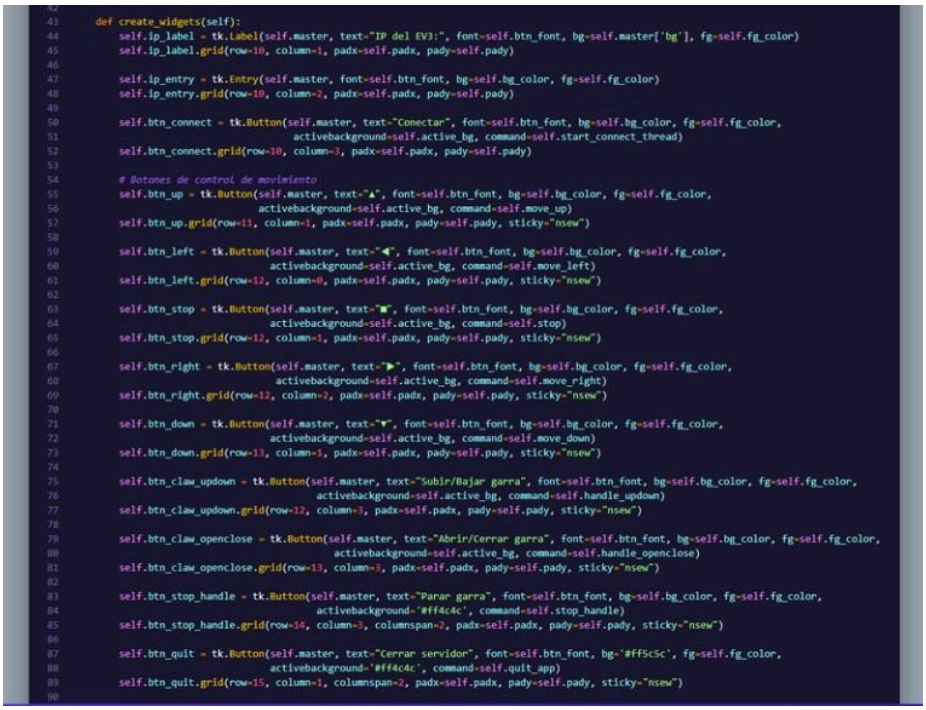
-Servidor

```python
import socket
from library import *

# Creacn del socket
s = socket.socket()
print("Socket creado")

 # Definicn del puerto
port = 8080
s.bind(('192.168.70.191', port))
print("El socket se crcon puerto: {}".format(port))

# Escuchando conexiones
s.listen(5)
print("El socket esescuchando...")

# Aceptacin de la conexn
connect, addr = s.accept()
print("Se conecta {}".format(addr))

# Bucle principal para recibir datos
while True:
    rawByte = connect.recv(1)
    char = rawByte.decode('utf-8')
```

```python
import socket
from library import *

# Creacn del socket
s = socket.socket()
print("Socket creado")

 # Definicn del puerto
port = 8080
s.bind(('192.168.70.191', port))
print("El socket se crcon puerto: {}".format(port))

# Escuchando conexiones
s.listen(5)
print("El socket esescuchando...")

# Aceptacin de la conexn
connect, addr = s.accept()
print("Se conecta {}".format(addr))

# Bucle principal para recibir datos
while True:
    rawByte = connect.recv(1)
    char = rawByte.decode('utf-8')
```

-Interfaz gráfica

```python
import socket
import tkinter as tk
from tkinter import font, messagebox
import threading

class EV3ControlApp:
    contador_subir = 0
    contador_bajar = 1
    contador_abrir = 1
    contador_cerrar = 0

    def __init__(self, master):
        self.master = master
        self.master.title("EV3 Control Panel")
        self.master.configure(bg='#1a1a1a')
        self.master.geometry('600x400')

        self.sock = None

        # Configuración de columnas y filas
        self.master.grid_columnconfigure(0, weight=1)
        self.master.grid_columnconfigure(1, weight=1)
        self.master.grid_columnconfigure(2, weight=1)
        self.master.grid_columnconfigure(3, weight=1)
        self.master.grid_rowconfigure(10, weight=1)
        self.master.grid_rowconfigure(11, weight=1)
        self.master.grid_rowconfigure(12, weight=1)
        self.master.grid_rowconfigure(13, weight=1)
        self.master.grid_rowconfigure(14, weight=1)
        self.master.grid_rowconfigure(15, weight=1)

        # Configuración de fuentes y estilo
        self.btn_font = font.Font(family='Helvetica', size=14,
    weight='bold')
        self.bg_color = '#2e2e2e'
        self.fg_color = 'white'
        self.active_bg = '#4d4d4d'
        self.padx = 15
        self.pady = 15

        # Crear widgets
        self.create_widgets()
```

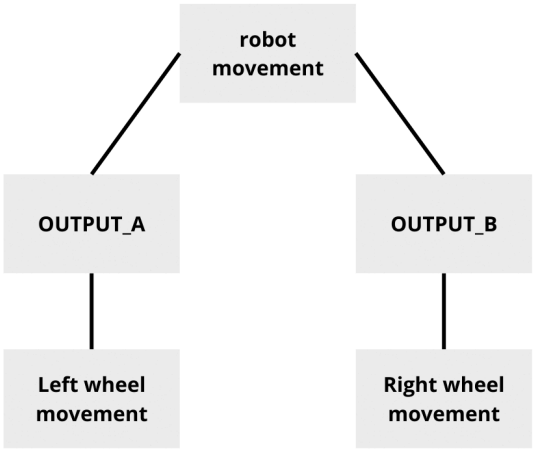


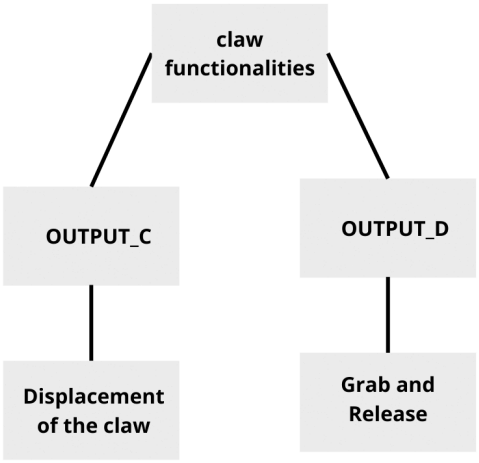
```python
    def create_widgets(self):
        self.ip_label = tk.Label(self.master, text="IP del EV3:", font=self.btn_font, bg=self.master['bg'], fg=self.fg_color)
        self.ip_label.grid(row=10, column=1, padx=self.padx, pady=self.pady)

        self.ip_entry = tk.Entry(self.master, font=self.btn_font, bg=self.bg_color, fg=self.fg_color)
        self.ip_entry.grid(row=10, column=2, padx=self.padx, pady=self.pady)

        self.btn_connect = tk.Button(self.master, text="Conectar", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                     activebackground=self.active_bg, command=self.start_connect_thread)
        self.btn_connect.grid(row=10, column=3, padx=self.padx, pady=self.pady)

        # Botones de control de movimiento
        self.btn_up = tk.Button(self.master, text="▲", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                activebackground=self.active_bg, command=self.move_up)
        self.btn_up.grid(row=11, column=1, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_left = tk.Button(self.master, text="◀", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                  activebackground=self.active_bg, command=self.move_left)
        self.btn_left.grid(row=12, column=0, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_stop = tk.Button(self.master, text="■", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                  activebackground=self.active_bg, command=self.stop)
        self.btn_stop.grid(row=12, column=1, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_right = tk.Button(self.master, text="▶", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                   activebackground=self.active_bg, command=self.move_right)
        self.btn_right.grid(row=12, column=2, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_down = tk.Button(self.master, text="▼", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                  activebackground=self.active_bg, command=self.move_down)
        self.btn_down.grid(row=13, column=1, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_claw_updown = tk.Button(self.master, text="Subir/Bajar garra", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                         activebackground=self.active_bg, command=self.handle_updown)
        self.btn_claw_updown.grid(row=12, column=3, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_claw_openclose = tk.Button(self.master, text="Abrir/Cerrar garra", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                            activebackground=self.active_bg, command=self.handle_openclose)
        self.btn_claw_openclose.grid(row=13, column=3, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_stop_handle = tk.Button(self.master, text="Parar garra", font=self.btn_font, bg=self.bg_color, fg=self.fg_color,
                                         activebackground='#ff4c4c', command=self.stop_handle)
        self.btn_stop_handle.grid(row=14, column=3, columnspan=2, padx=self.padx, pady=self.pady, sticky="nsew")

        self.btn_quit = tk.Button(self.master, text="Cerrar servidor", font=self.btn_font, bg='#ff5c5c', fg=self.fg_color,
                                  activebackground='#ff4c4c', command=self.quit_app)
        self.btn_quit.grid(row=15, column=1, columnspan=2, padx=self.padx, pady=self.pady, sticky="nsew")
```

```
    def start_connect_thread(self):
        """Inicia un hilo para la conexión, manteniendo la interfaz responsiva."""
        ip = self.ip_entry.get()
        if self.is_valid_ip(ip):
            threading.Thread(target=self.connect, args=(ip,), daemon=True).start()
        else:
            messagebox.showerror("Error de entrada", "La dirección IP ingresada no es válida. Por favor, intente de nuevo.")

    def is_valid_ip(self, ip):
        """Verifica si la IP ingresada es válida."""
        parts = ip.split('.')
        if len(parts) != 4:
            return False
        for part in parts:
            if not part.isdigit() or not (0 <= int(part) < 256):
                return False
        return True

    def connect(self, ip):
        try:
            if self.sock is not None:
                self.sock.close()
            self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            self.sock.connect((ip, 8888))
            messagebox.showinfo("Conexión", "Conectado correctamente al EV3.")
            self.enable_controls(True)
        except Exception as e:
            messagebox.showerror("Error de conexión", f"No se pudo conectar a {ip}. \n{str(e)}")
            self.enable_controls(False)

    def enable_controls(self, state):
        """Habilita o deshabilita los botones de control según el estado proporcionado."""
        self.btn_up.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_left.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_stop.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_right.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_down.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_claw_updown.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_claw_openclose.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_stop_handle.config(state=tk.NORMAL if state else tk.DISABLED)
        self.btn_quit.config(state=tk.NORMAL if state else tk.DISABLED)

    def send_command(self, command):
        if self.sock:
            self.sock.sendall(command.encode('utf-8'))

    def move_up(self):
        self.send_command('w')
```

```
    def move_down(self):
        self.send_command('s')

    def move_left(self):
        self.send_command('a')

    def move_right(self):
        self.send_command('d')

    def stop(self):
        self.send_command('x')

    def handle_updown(self):
        if EV3ControlApp.contador_bajar == 0:
            self.send_command('g')
            EV3ControlApp.contador_bajar += 1
            EV3ControlApp.contador_subir = 0
        else:
            self.send_command('t')
            EV3ControlApp.contador_subir += 1
            EV3ControlApp.contador_bajar = 0

    def handle_openclose(self):
        if EV3ControlApp.contador_cerrar == 0:
            self.send_command('f')
            EV3ControlApp.contador_cerrar += 1
            EV3ControlApp.contador_abrir = 0
        else:
            self.send_command('r')
            EV3ControlApp.contador_abrir += 1
            EV3ControlApp.contador_cerrar = 0

    def stop_handle(self):
        self.send_command('c')

    def quit_app(self):
        if EV3ControlApp.contador_abrir == 0:
            self.send_command('r')
        if EV3ControlApp.contador_bajar == 0:
            self.send_command('g')
        self.send_command('c')
        self.send_command('q')
        self.sock.close()
        self.master.quit()

if __name__ == "__main__":
    root = tk.Tk()
    app = EV3ControlApp(root)
    root.mainloop()
```

6.3. Diagramas





7. Resultados

7.1. Estado actual del proyecto

Hasta el momento, el proyecto ha logrado cumplir con los siguientes aspectos::

- Armado y diseño del robot: El robot EV3 ya tiene su diseño establecido y su armado está listo, considerado la versión final.
- Funciones de movimiento y agarre: El robot EV3 ya tiene sus funciones de desplazamiento y agarre completamente operativas y funcionando a la perfección.
- Interfaz gráfica Tk: La interfaz gráfica encargada de controlar las funcionalidades del EV3 se encuentra finalizada y se ejecuta correctamente.
- Conexión remota: La conexión remota con el robot ha sido establecida.

7.2. Problemas encontrados y solución propuesta

Problemas	Soluciones
Los archivos almacenados en el robot EV3 se desvanecen sin explicación aparente	Para evitar la pérdida de archivos, estos se almacenan en un pendrive.
Cambios regulares de la estructura del robot, cambiando su diseño casi en totalidad.	Buscar un diseño que pueda llevar a cabo sus funciones necesarias, como agarre y movilidad, de manera óptima y eficiente.
Escasez de piezas para el ensamblado del EV3	Se le pidieron las piezas faltantes a los ayudantes

8. Conclusión

El proyecto "EVA-π3" representa un desafío multidisciplinario que ha permitido a los estudiantes aplicar y consolidar conocimientos en ingeniería, programación, matemática y diseño, dentro de un contexto práctico y colaborativo. A través de la construcción y programación del robot, se han integrado tareas complejas de diseño mecánico, desarrollo de software y gestión de proyectos, utilizando herramientas como GitHub y Visual Studio Code y sistemas operativos robustos como Linux.

El equipo ha demostrado una excelente capacidad para adaptarse a desafíos técnicos y organizativos, manteniendo siempre un enfoque en la consecución de los objetivos del proyecto. Se destaca el compromiso y la sinergia entre los miembros del equipo, elementos que han sido fundamentales para el avance continuo del proyecto.

De cara al futuro, se plantean posibles líneas de investigación y desarrollo adicionales en el ámbito de la robótica educativa, incluyendo la integración de algoritmos de inteligencia artificial para mejorar la autonomía del robot. Esta experiencia ofrece una base sólida para futuros proyectos, enriqueciendo el conocimiento y las competencias de los estudiantes para su carrera académica y profesional.

9. Referencias

LEGO Education: Education EV3 Expansion Set (45560) 673419196123 | eBay

Precios de Canva: compara los planes Gratis, Pro, Equipos y Empresas

Lenovo L340 | MercadoLibre 📦

Touchscreen Toshiba Tecra Z40T-C i5-6300U 8GB 240GB SSD 1920x1080 Clase A Windows 10 Home.

Dynabook - Tecra Z40 Series Specifications

LEGO Mindstorms 45544 EV3 STEM Core Set Complete Battery And Charger READ Det | eBay

https://licenciasdigitales.cl/product/microsoft-office-2021-professional-plus-permanente/?gad_source=1&gclid=Cj0KCQjwlvW2BhDyARIsADnle-JYwn57JD79O3yVVhDbxmGCj8C7FPp3sXXIcTgxXB2VzT-Q59sSZ4UaAr-EEALw_wcB