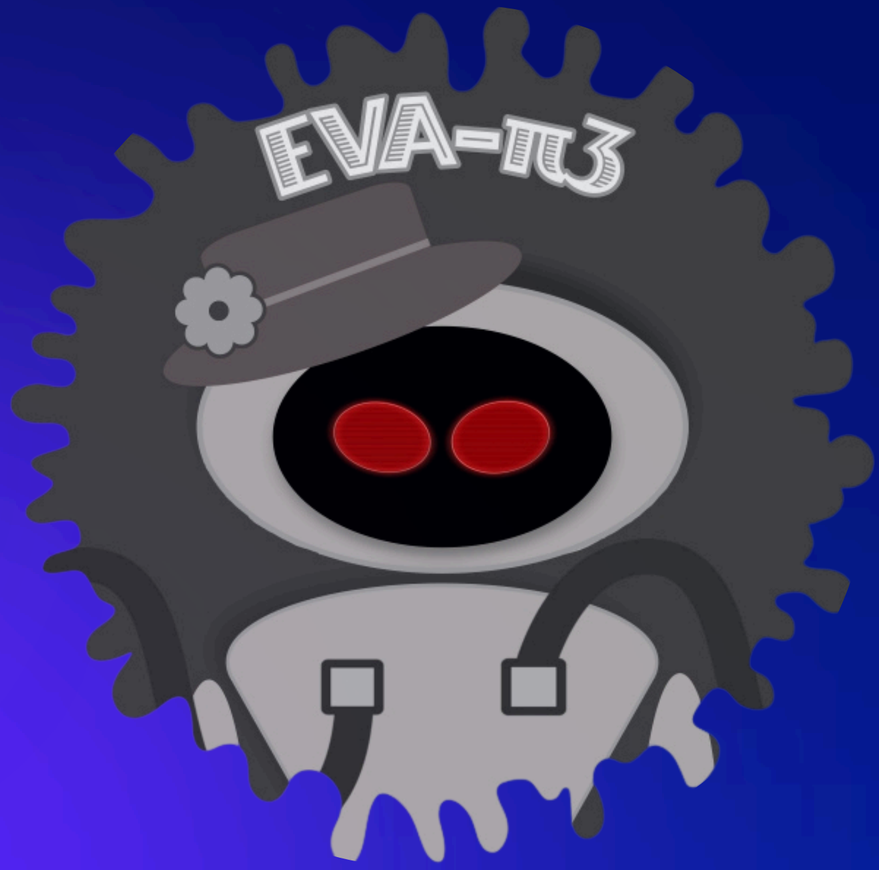
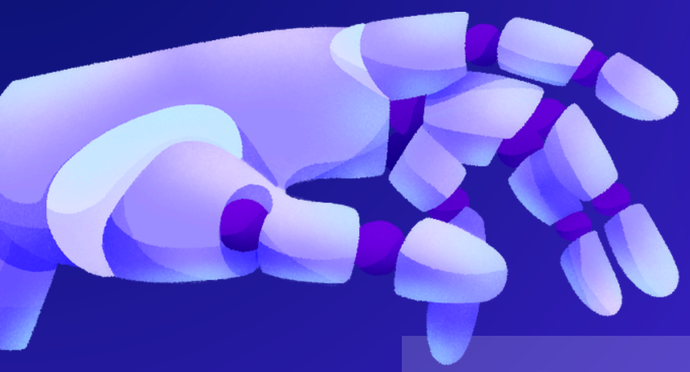
PROYECTO 1

EVA-П3

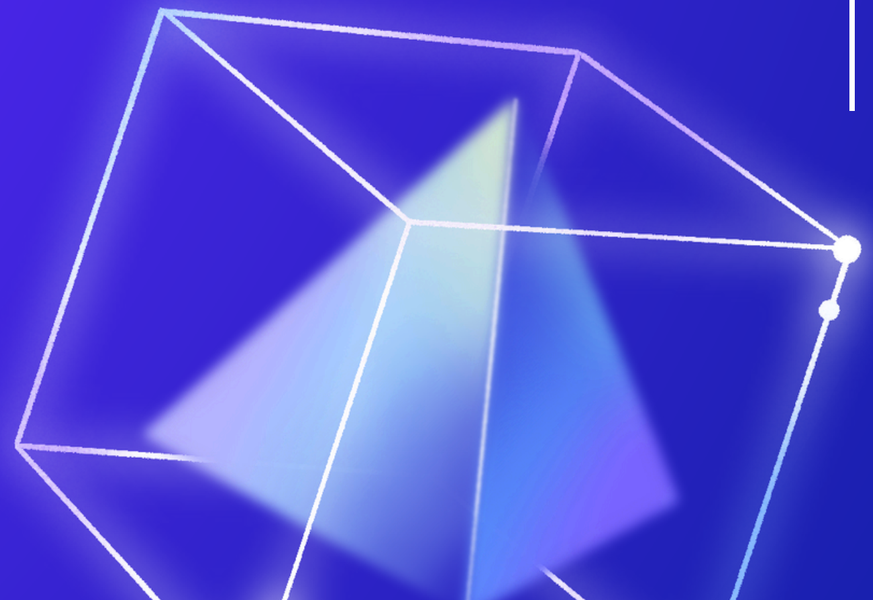


Integrantes:
Antonella Butrón
Sebastian Eyraud
Benjamin Tamarin
Josue Sucso
Bastián Cruz

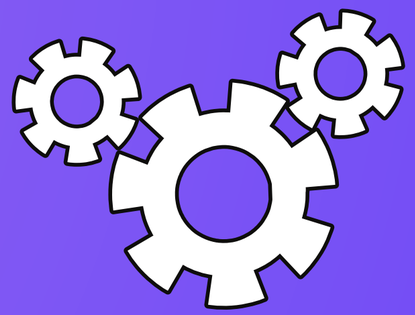
INTRODUCCIÓN



ÍNDICE



PERSONAL



Ensamblador

Sebastian Eyraud

Diseñador

Sebastian Eyraud

Jefe de proyecto

Benjamin Tamarin

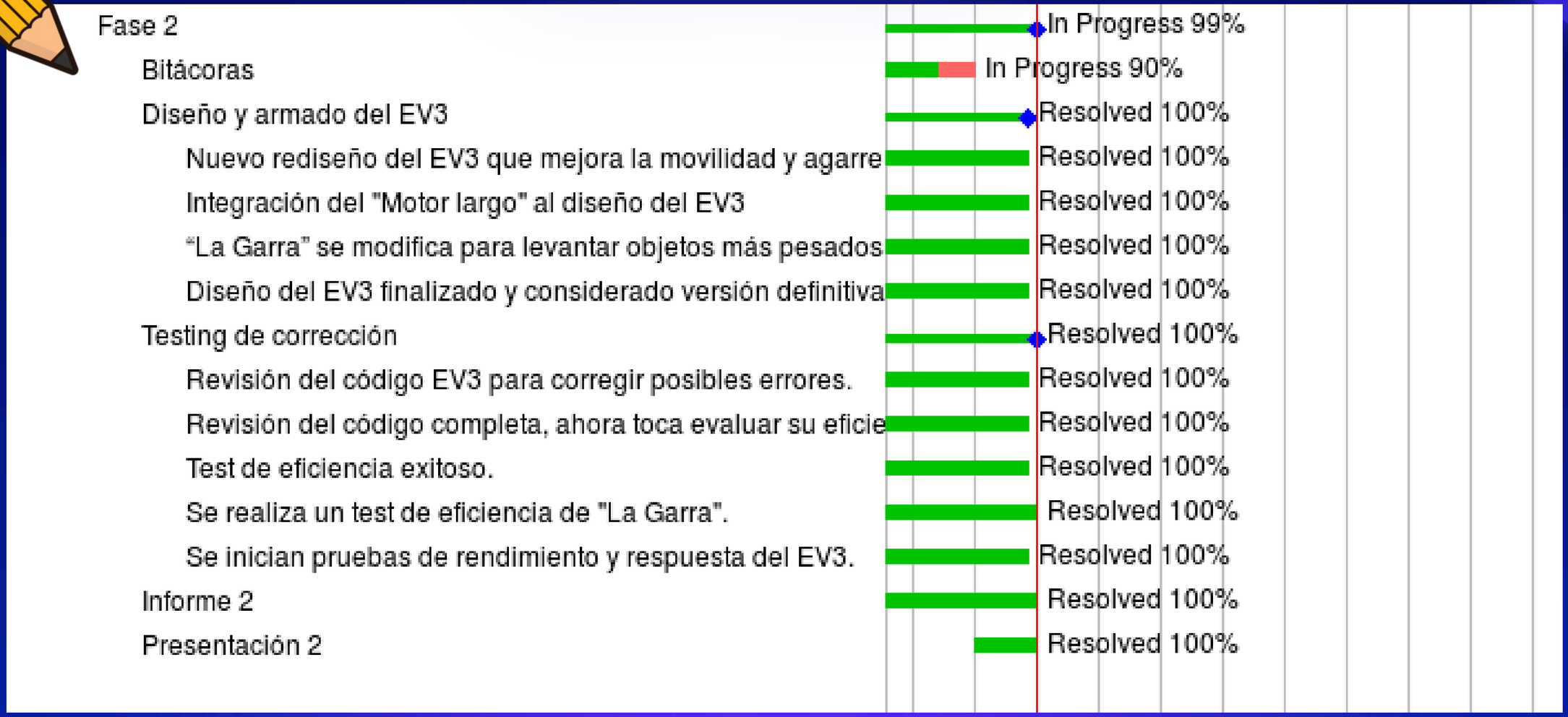
Programador

Benjamin Tamarin
Josue Suso

Documentador

Antonella Butrón
Bastián Cruz

CARTA GANTT


Fase 2
In Progress 99%
Bitácoras
In Progress 90%
Diseño y armado del EV3
Resolved 100%
Nuevo rediseño del EV3 que mejora la movilidad y agarre
Resolved 100%
Integración del "Motor largo" al diseño del EV3
Resolved 100%
“La Garra” se modifica para levantar objetos más pesados
Resolved 100%
Diseño del EV3 finalizado y considerado versión definitiva
Resolved 100%
Testing de corrección
Resolved 100%
Revisión del código EV3 para corregir posibles errores.
Resolved 100%
Revisión del código completa, ahora toca evaluar su eficie
Resolved 100%
Test de eficiencia exitoso.
Resolved 100%
Se realiza un test de eficiencia de "La Garra".
Resolved 100%
Se inician pruebas de rendimiento y respuesta del EV3.
Resolved 100%
Informe 2
Resolved 100%
Presentación 2
Resolved 100%

Requerimientos

Funcionales
1-El robot debe poder moverse en cualquier dirección.
2-El acceso remoto del robot debe funcionar adecuadamente mediante una interfaz gráfica.
3-El robot debe poder procesar las órdenes enviadas desde el servidor de manera eficiente y en tiempo real.

Requerimientos

No funcionales
1. 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.
2. El código del sistema debe estar bien documentado para facilitar actualizaciones futuras.
3.. El proyecto debe contar con un manual de usuario que sirva de guia para el uso del robot y su interfaz.

Arquitectura del proyecto

1

Conexión

2

Servidor

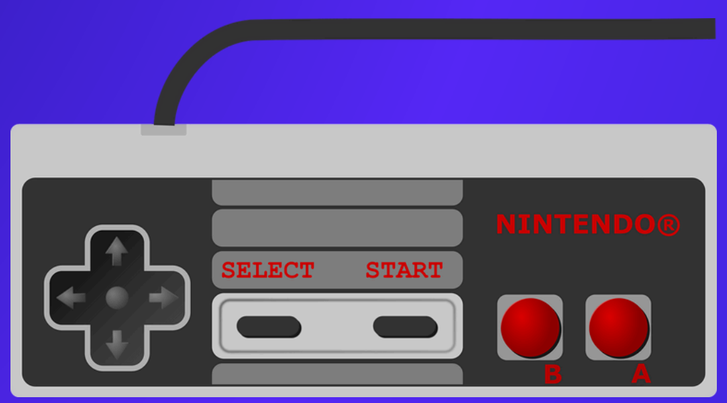
3

Cliente

4

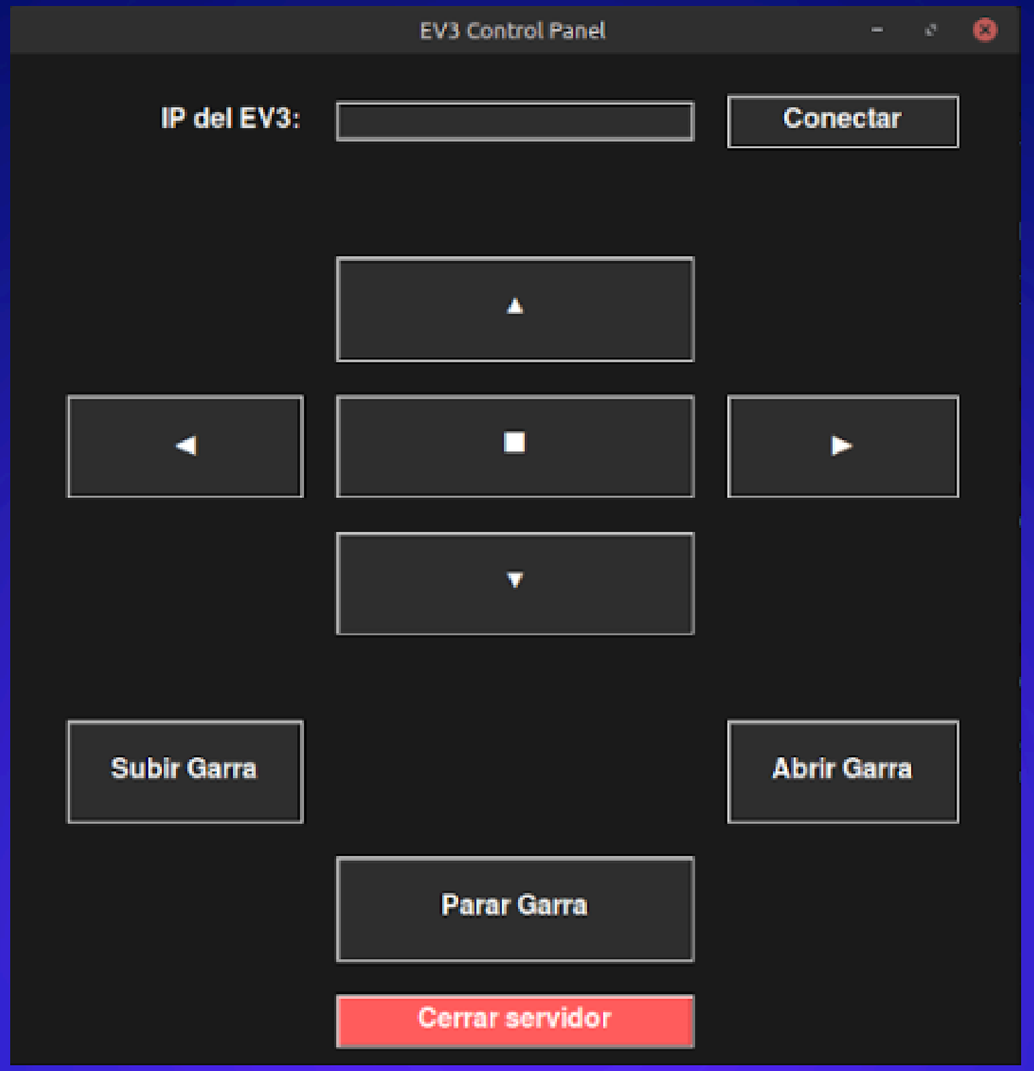
Robot

5



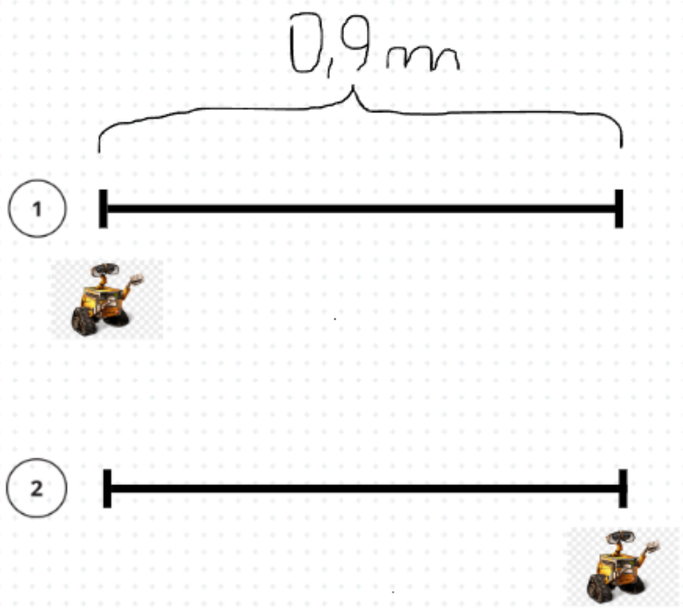
Interfaz grafica

Interfaz


EV3 Control Panel
IP del EV3:
Conectar
▲
◀
■
▶
▼
Subir Garra
Abrir Garra
Parar Garra
Cerrar servidor

Fundamentos de las acciones del Robot

Velocidad de desplazamiento del robot



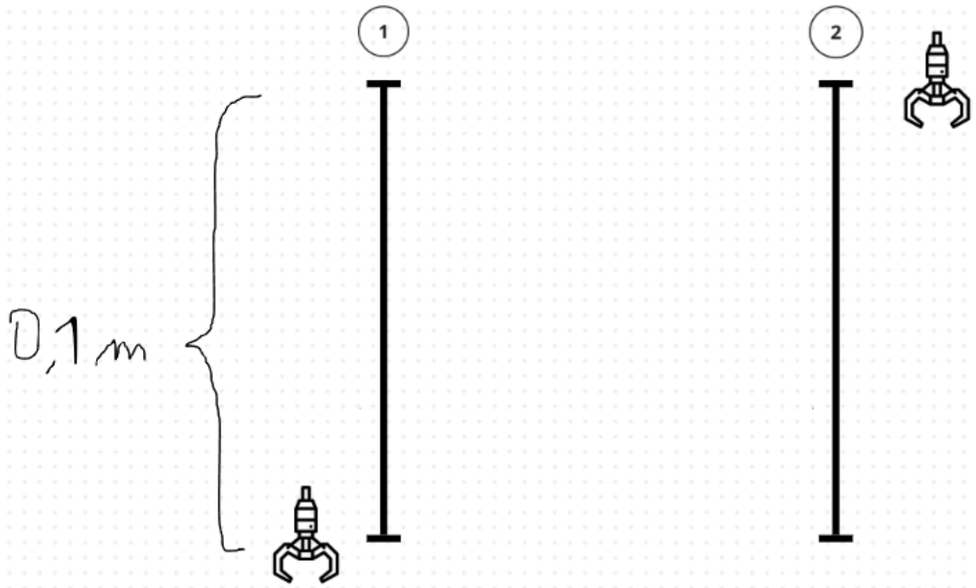
Desplazamiento del robot(d): 0.90m

Tiempo de desplazamiento(t): 10.56s

Velocidad de desplazamiento del robot (v):

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

Velocidad de elevación de la Garra



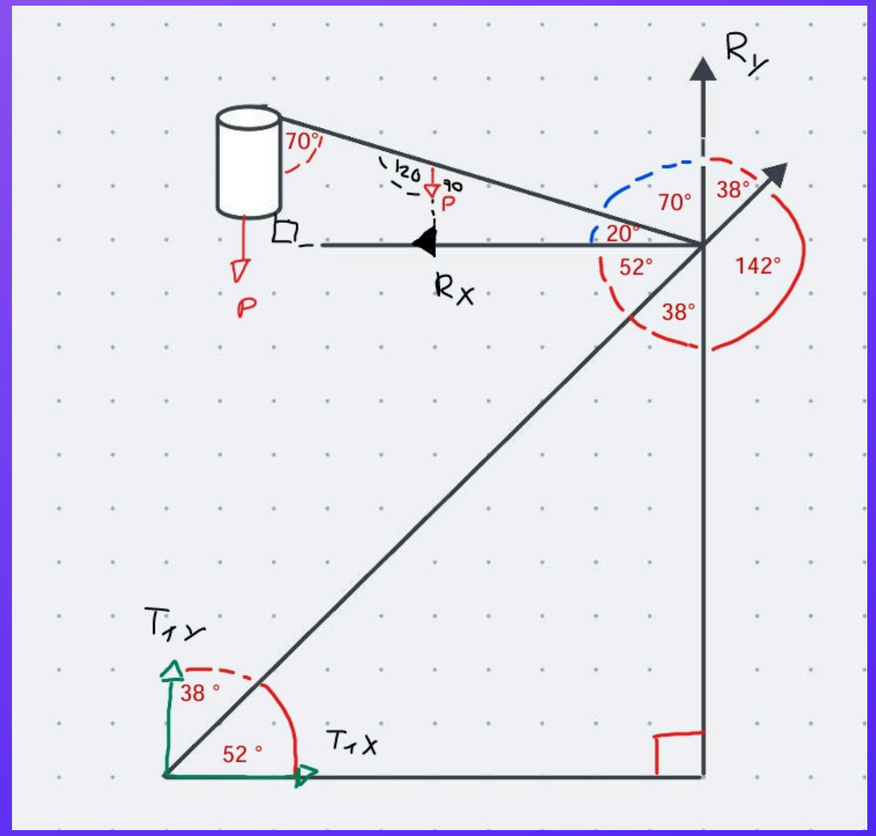
Elevación de la Garra(h): 0.1m

Tiempo de elevación(t): 2s

Velocidad de elevación de la Garra (v):

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

Movimiento Torque Estático



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

Descripción de los Programas y Diagrama

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
# Movimiento hacia arriba
    if char == 'w':
        move_up()

    # Movimiento hacia abajo
    if char == 's':
        move_down()

    # Mover a la derecha
    if char == 'd':
        move_right()

    # Mover a la izquierda
    if char == 'a':
        move_left()

    if char == 't':
     handle_up()

    if char == 'g':
     handle_down()

    if char == 'r':
     handle_open()

    if char == 'f':
     handle_close()

    if char == 'c':
        stop_handle()

    if char == 'x':
        stop()

    if char == 'q':
        print("Terminando la sesn...")
        break
```

INTERFAZ GRAFICA Tk

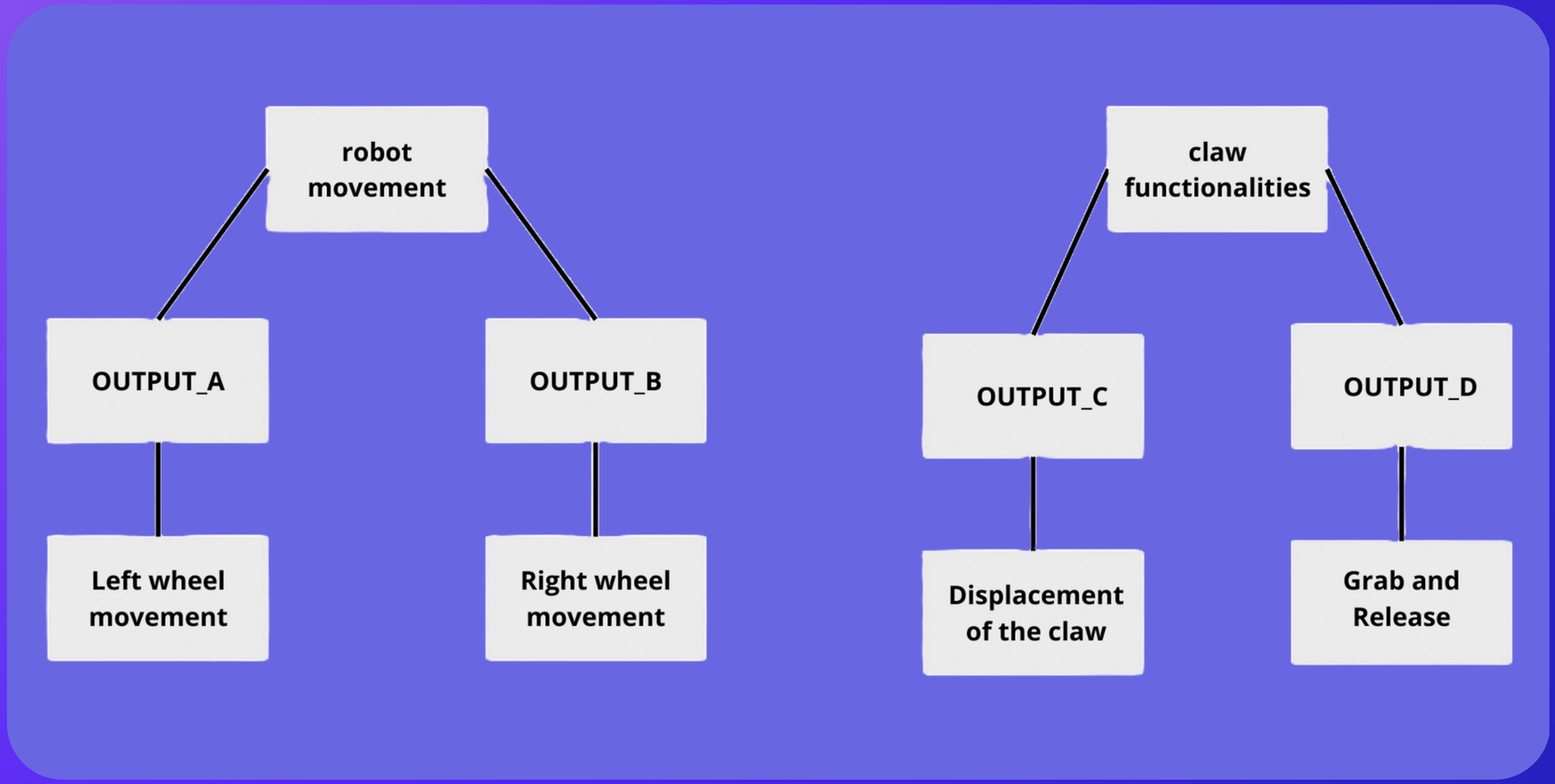
```python
import socket
import tkinter as tk
from tkinter import font, messagebox
import threading

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()
```

```
    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")

```

```python

    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, 8080))
            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')
```

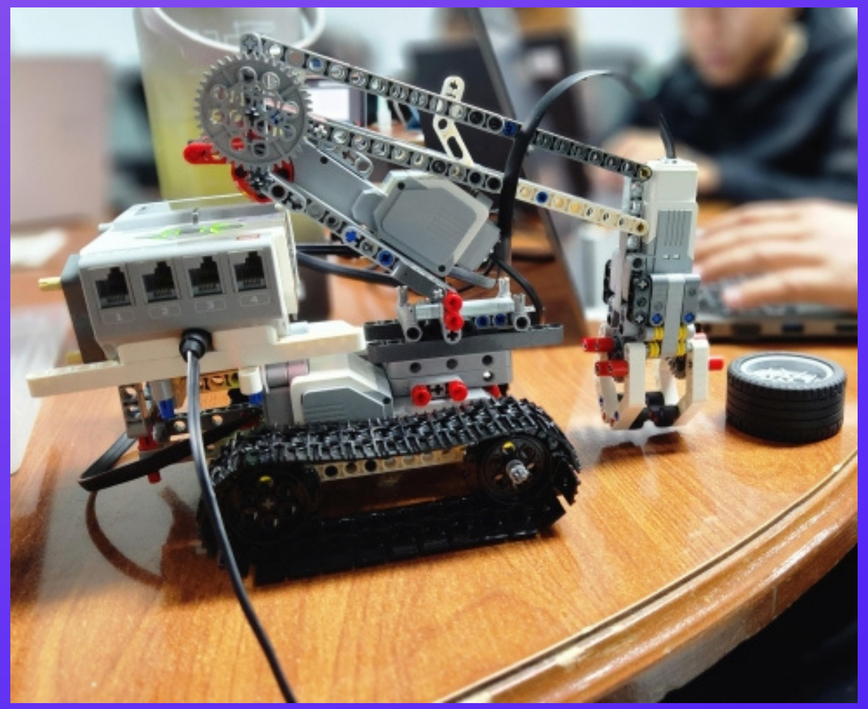
```python
    def move_down(self):
        self.send_command('s')

    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()
```

DIAGRAMA


robot movement
OUTPUT_A
OUTPUT_B
Left wheel movement
Right wheel movement
claw functionalities
OUTPUT_C
OUTPUT_D
Displacement of the claw
Grab and Release

Estado Actual

Versión final



Diseño y armado del robot:

Ejecución de funciones de movimiento y agarre:

Interfaz gráfica Tk:

PROBLEMAS Y SOLUCIONES

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

CONCLUSIÓN

¡MUCHAS GRACIAS!