# TERRENEITOR

## TODO TERRENO

INTEGRANTES:


RAFAEL NAKATA
TOMÁS CARVAJAL
ARIEL COLQUE
BRANDON JALANOCA
JOAQUÍN JELVES


START

MENU

# TABLA DE CONTENIDO

## TEMAS ABORDADOS

# INTRODUCCIÓN

- DURANTE LA PRESENTACIÓN, VEREMOS LOS PROGRESOS REALIZADOS TANTO EN EL DISEÑO DEL ROBOT COMO EN SU CÓDIGO, Y CÓMO ESTE SE MODIFICÓ DEBIDO A ALGUNOS INCONVENIENTES QUE HAN APARECIDO EN EL CAMINO. TODO ESTO PARA QUE PUEDA LLEVAR A CABO LA FUNCIÓN PARA LA QUE FUE DISEÑADO, DESPLAZARSE Y TRANSPORTAR UNA PELOTA DE PING PONG UTILIZANDO UNA GARRA.

# EQUIPO

## ANTES

JEFE DE GRUPO: RAFAEL NAKATA

ENSAMBLADOR: TOMÁS CARVAJAL, JOAQUÍN JELVES

DOCUMENTADOR: BRANDON JALANOCA

PROGRAMADOR: RAFAEL NAKATA, ARIEL COLQUE, BRANDON JALANOCA

DISEÑADOR: TOMÁS CARVAJAL, JOAQUÍN JELVES, RAFAEL NAKATA

## DESPUÉS

JEFE DE GRUPO: RAFAEL NAKATA

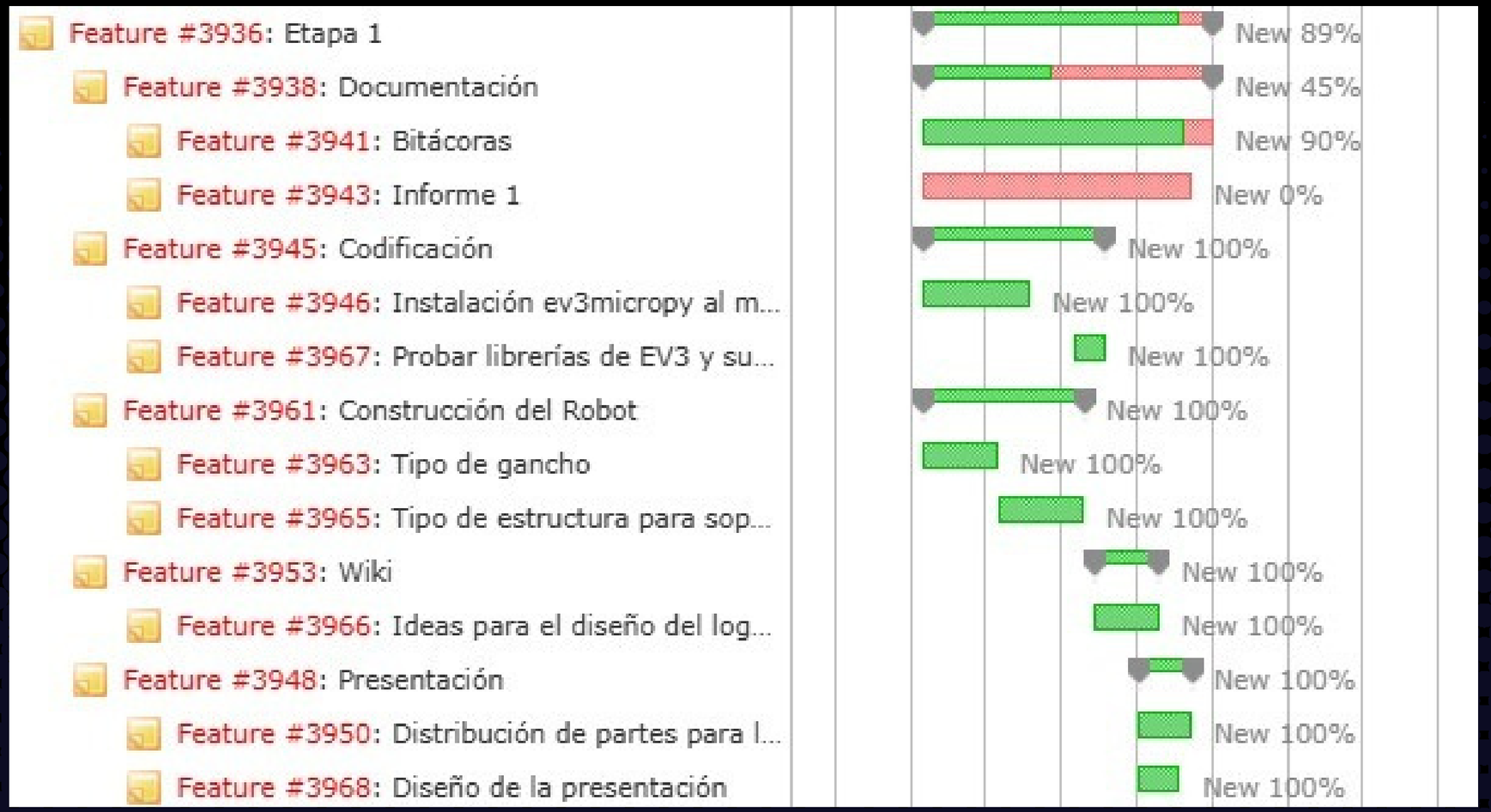
ENSAMBLADOR: TOMÁS CARVAJAL, JOAQUÍN JELVES

DOCUMENTADOR: BRANDON JALANOCA, ARIEL COLQUE

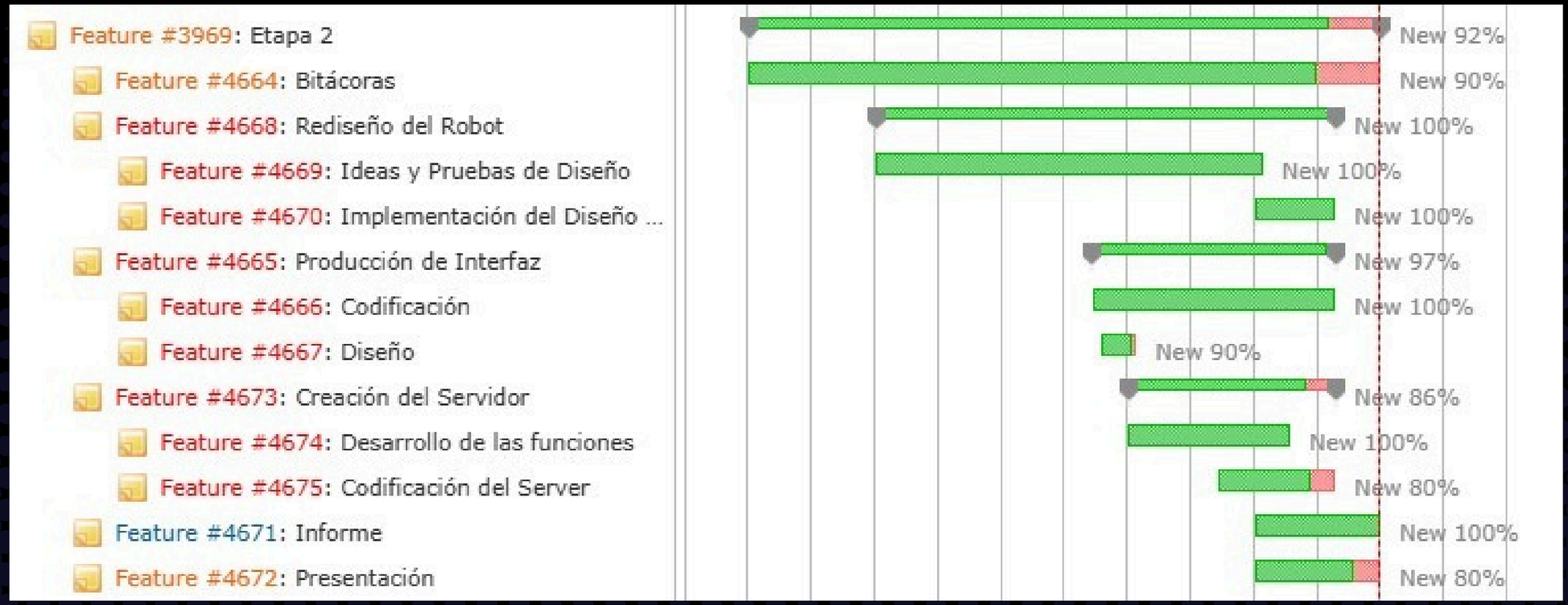
PROGRAMADOR: RAFAEL NAKATA

DISEÑADOR: TOMÁS CARVAJAL, JOAQUÍN JELVES, RAFAEL NAKATA

# CARTA GANTT


Feature #3936: Etapa 1
Feature #3938: Documentación
Feature #3941: Bitácoras
Feature #3943: Informe 1
Feature #3945: Codificación
Feature #3946: Instalación ev3micropy al m...
Feature #3967: Probar librerías de EV3 y su...
Feature #3961: Construcción del Robot
Feature #3963: Tipo de gancho
Feature #3965: Tipo de estructura para sop...
Feature #3953: Wiki
Feature #3966: Ideas para el diseño del log...
Feature #3948: Presentación
Feature #3950: Distribución de partes para l...
Feature #3968: Diseño de la presentación
New 89%
New 45%
New 90%
New 0%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%

# CARTA GANTT


Feature #3969: Etapa 2
Feature #4664: Bitácoras
Feature #4668: Rediseño del Robot
Feature #4669: Ideas y Pruebas de Diseño
Feature #4670: Implementación del Diseño ...
Feature #4665: Producción de Interfaz
Feature #4666: Codificación
Feature #4667: Diseño
Feature #4673: Creación del Servidor
Feature #4674: Desarrollo de las funciones
Feature #4675: Codificación del Server
Feature #4671: Informe
Feature #4672: Presentación
New 92%
New 90%
New 100%
New 100%
New 100%
New 97%
New 100%
New 90%
New 86%
New 100%
New 80%
New 100%
New 80%

# REQUERIMIENTOS

## FUNCIONALES

- EL ROBOT DEBE SER MANEJADO A TRAVÉS DE UNA INTERFAZ GRÁFICA.
- EL ROBOT DEBE TENER LA CAPACIDAD DE DESPLAZARSE.
- EL ROBOT DEBE TENER LA CAPACIDAD DE AGARRAR UNA PELOTA DE PING PONG.

## NO FUNCIONALES

- LA INTERFAZ GRÁFICA DEBE CONTENER LOS MOVIMIENTOS DEL ROBOT.
- LA ESTRUCTURA DEL ROBOT DEBE SER ESTABLE Y CAPAZ DE LLEVAR A CABO TODO TIPO DE ACCIONES DE MOVIMIENTO.

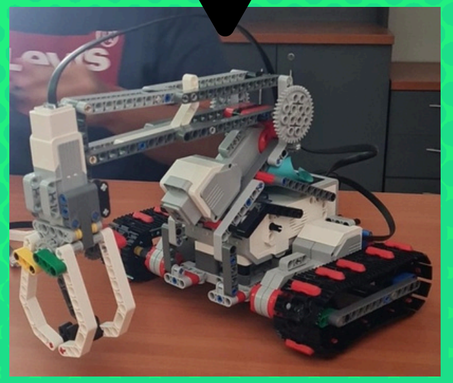
# ARQUITECTURA

INTERNET

CLIENTE

SERVIDOR


INTERFAZ



ROBOT

# ★INTERFAZ★

# ★CODIGO★

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.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)

# 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
```

# ★CODIGO★

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
```

# ★CODIGO★

```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))

```

```python
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)
```

```python
# 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()
```

# ESTADO ACTUAL

- FUNCIONES DE MOVIMIENTO IMPLEMENTADAS
- CONEXIÓN REMOTA ESTABLECIDA
- INTERFAZ GRAFICA DESARROLLADA
- REGISTRO DE ACTIVIDADES, INFORME Y PRESENTACIÓN
- SERVIDOR EN MARCHA Y FUNCIONANDO
- FINALIZACIÓN DE LA ULTIMA VERSIÓN DEL ROBOT POR EL MOMENTO

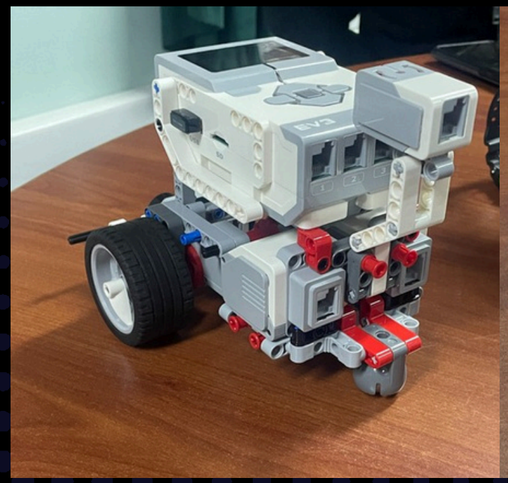
# PROBLEMAS

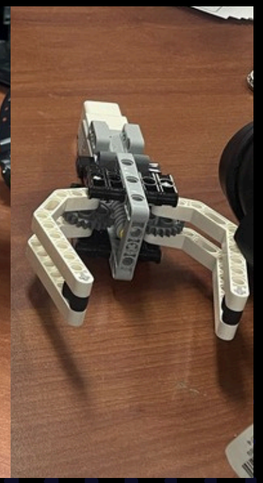
- PROBLEMAS CON LA ESTRUCTURA DE LA GARRA
- EL BLOQUE EV3 NO DETECTA LA TARJETA MICRO SD
- PROBLEMAS CON LA MOVILIDAD DE LA GARRA EN LA INTERFAZ
- PROBLEMAS CON LOS BOTONES DE LA INTERFAZ

# SOLUCIONES

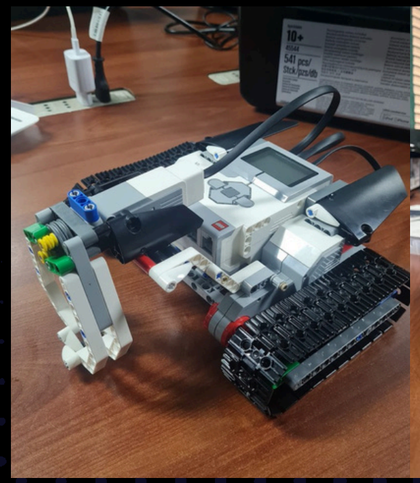
- RECONSTRUCCIÓN DE LA GARRA
- FORMATEAR LA MICRO SD
- REVISAR QUE ESTA FALLANDO EN EL CODIGO

# AVANCE DEL ROBOT

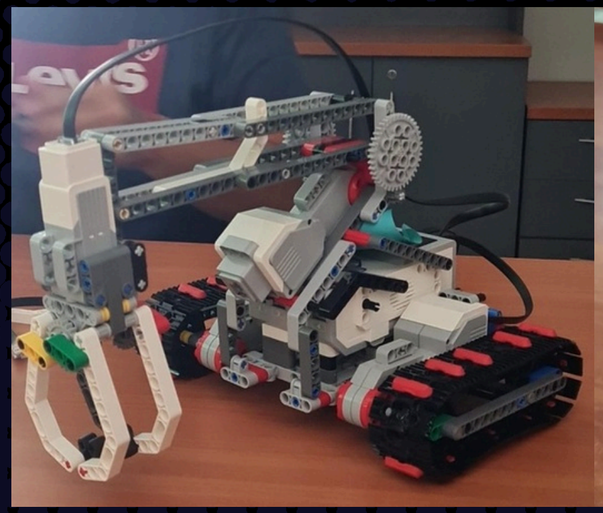




TERRE 1.0



TERRE 2.0



TERRE 3.0

# CONCLUSIÓN