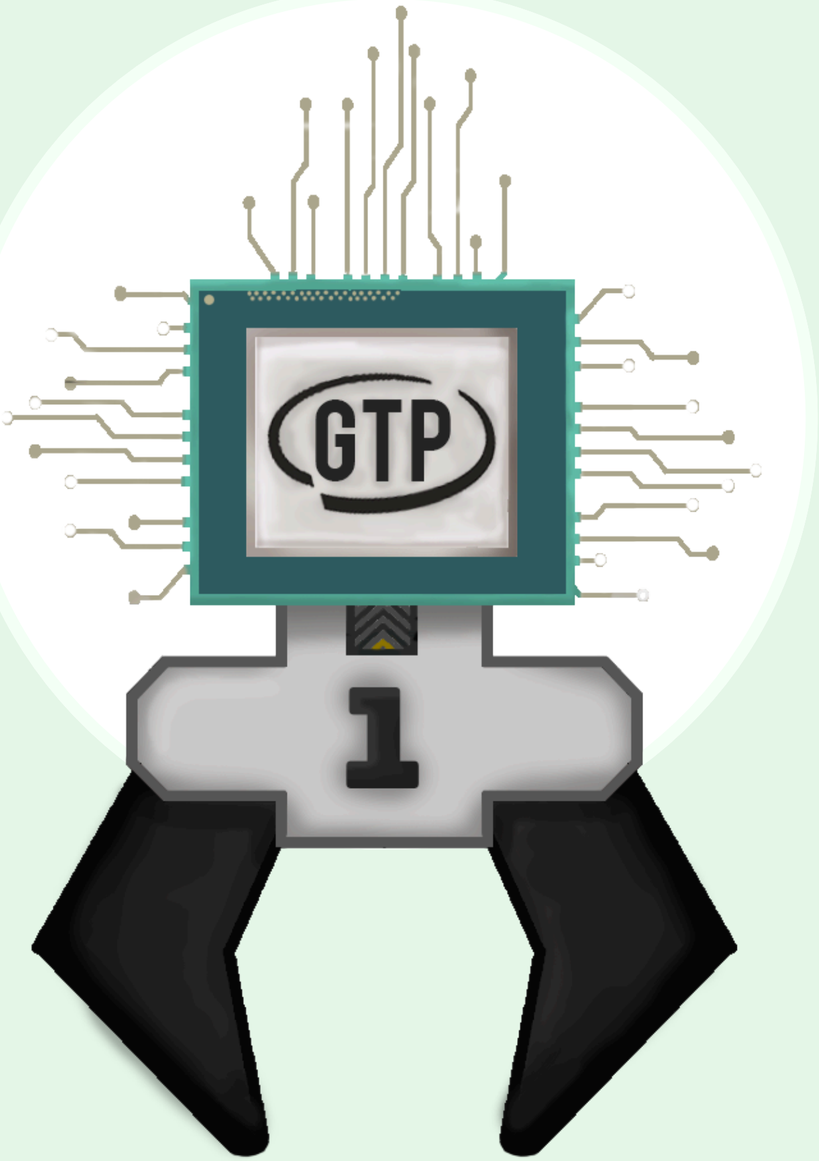
GTP-1

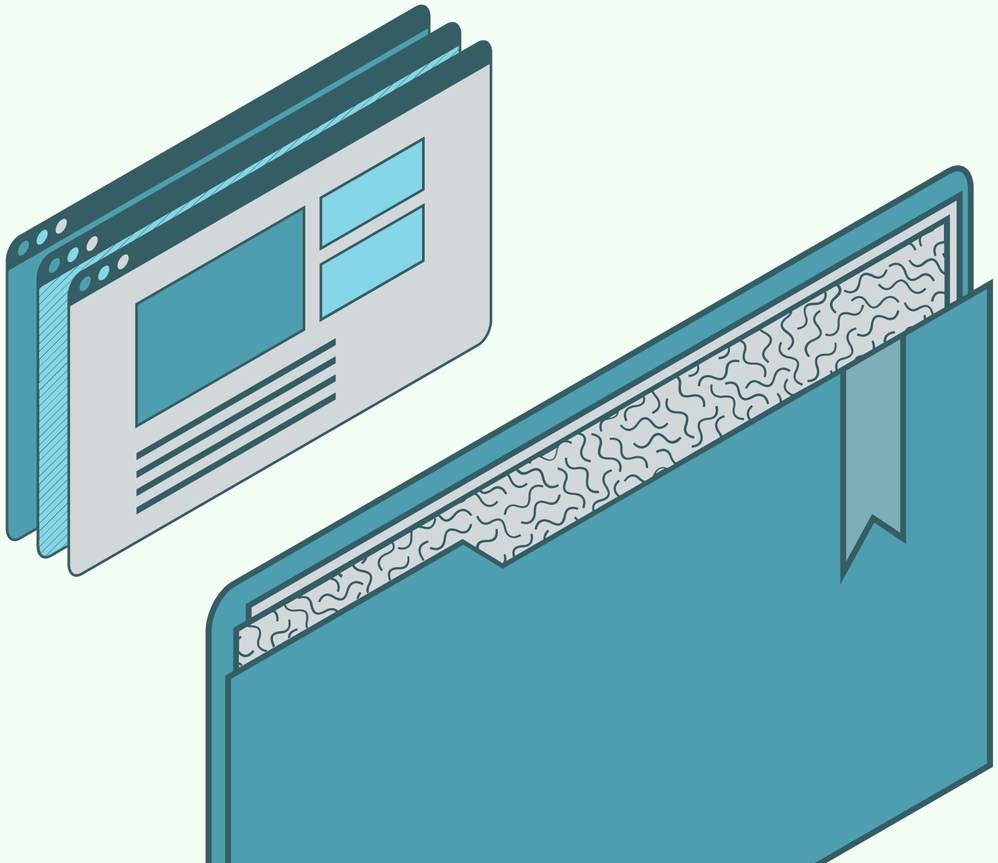


Integrantes:
Benjamin Flores
Alex Muñoz
Jonathan Orellana
Pattricio Medina

UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

Ingeniería
Computación e Informática

ÍNDICE



01. Panorama General

02. Organización del Personal

03. Planificación del Proyecto

04. Planificación de los Recursos

05. Análisis y Diseño

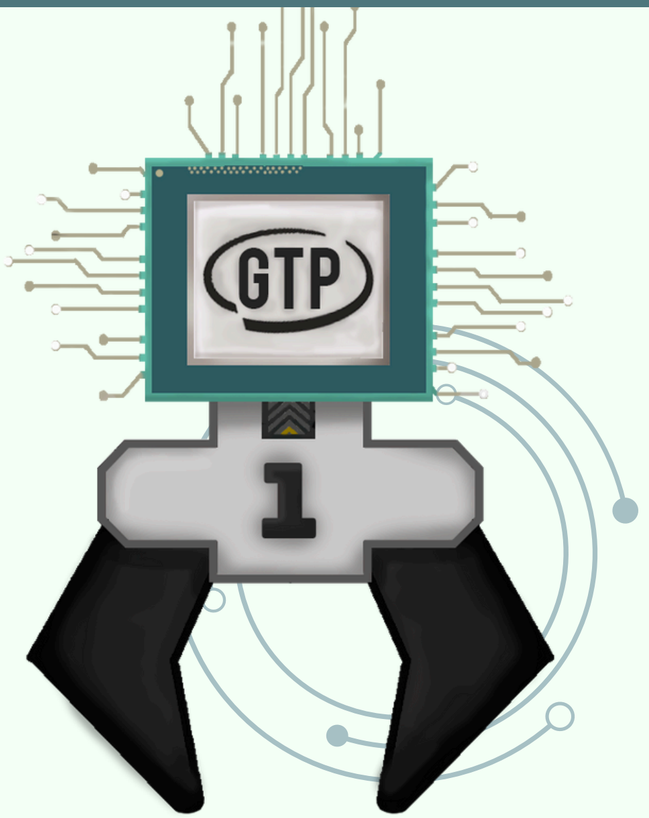
06. Implementación

07. Resultados

08. Conclusión

INTRODUCCIÓN

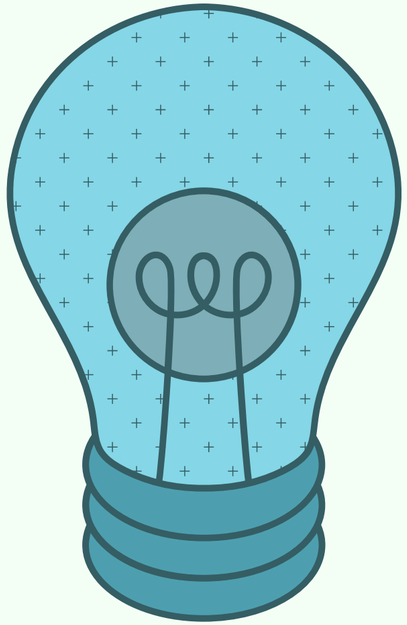
En el siguiente proyecto se descubrirá los procedimientos realizados para la construcción del robot Lego Ev3 la cual simularemos una garra de carga. El objetivo con el cual se espera conseguir con este proyecto es que el robot sea capaz de moverse con una interfaz y agarrar una pelota con una garra construida. Todos los algoritmos creados son introducidos en el programa Visual Studio Code con el lenguaje de Python, mediante un firmware llamado Ev3Dev que se instala dentro del robot mencionado anteriormente.



OBJETIVOS

GENERALES

- Desarrollar un plan de construcción de un robot Ev3, con el cual es codificado y encargado por estudiantes del departamento de Computación e informática.



ESPECIFICOS

- Planificar el tipo de robot a crear.
- Construir un robot con el kit EV3 Mindstorms.
- Instalar el firmware dentro del robot
- Realizar un software con algoritmos realizados con el lenguaje python para el movimiento del robot.
- Implementar una interfaz amigable para un manejo sencillo para el usuario.

RESTRICCIONES

- El lenguaje de programación utilizado dentro del Visual Studio Code será Python.
- Un límite de tiempo para la construcción del robot y su programación.
- Todos los documentos (Bitácoras, Carta Gantt e Informe) relacionados al proyecto deberán ser subidos a la plataforma Redmine.
- El sistema operativo utilizado será Linux.

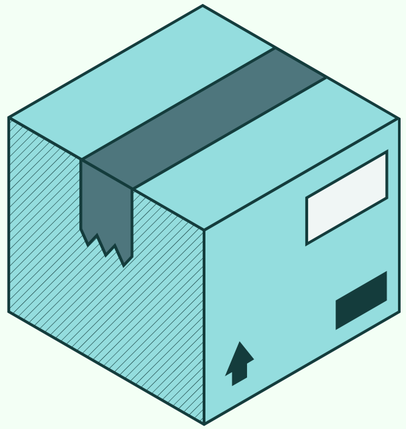
ENTREGABLES

BITÁCORAS

Bitácoras semanales entregadas todos los domingos.

INFORME

Informe de plan de proyecto.

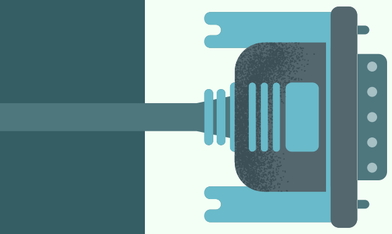


PRESENTACIÓN

Presentación de plan de proyecto.

CARTA GANTT

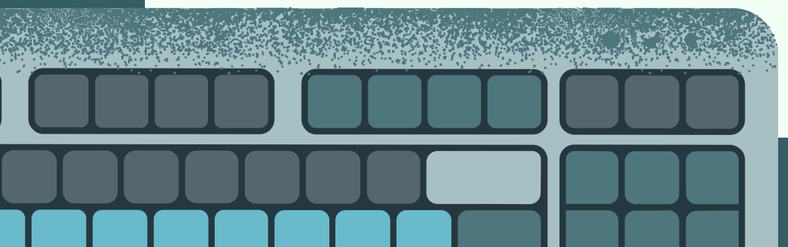
Es una representación gráfica de la planificación del proyecto.

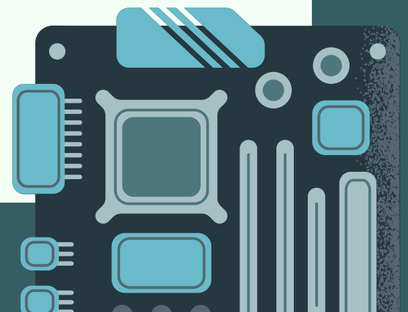


ACTUALIZACIÓN

Actualización del proyecto en Redmine.

Estos documentos, principalmente las bitácoras e informes, serán subidos a la plataforma Redmine de la Universidad





ORGANIZACIÓN

JEFE DE GRUPO

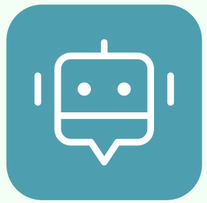
Alex Muñoz

El que representa al equipo de trabajo y mantener la organización e iniciativa del mismo.

PROGRAMADOR

Jonathan Orellana

Se encarga de crear e implementar los algoritmos necesarios para el movimiento y funcionamiento del robot.



ENSAMBLADOR

Alex Muñoz

Encargado del montaje y armado de robot al igual que monitora el cumplimiento de las funciones del robot.

DOCUMENTACIÓN

Benjamin Flores

Es el que se encargar en documentar todo lo que se realiza en el proyecto, los avances, informes y bitácoras y carta Gantt.

DISEÑADOR

Pattricio Medina

Encargado principalmente del diseño de: Logotipos, Presentaciones, Manual de usuario, etc....

MECANISMOS DE COMUNICACIÓN

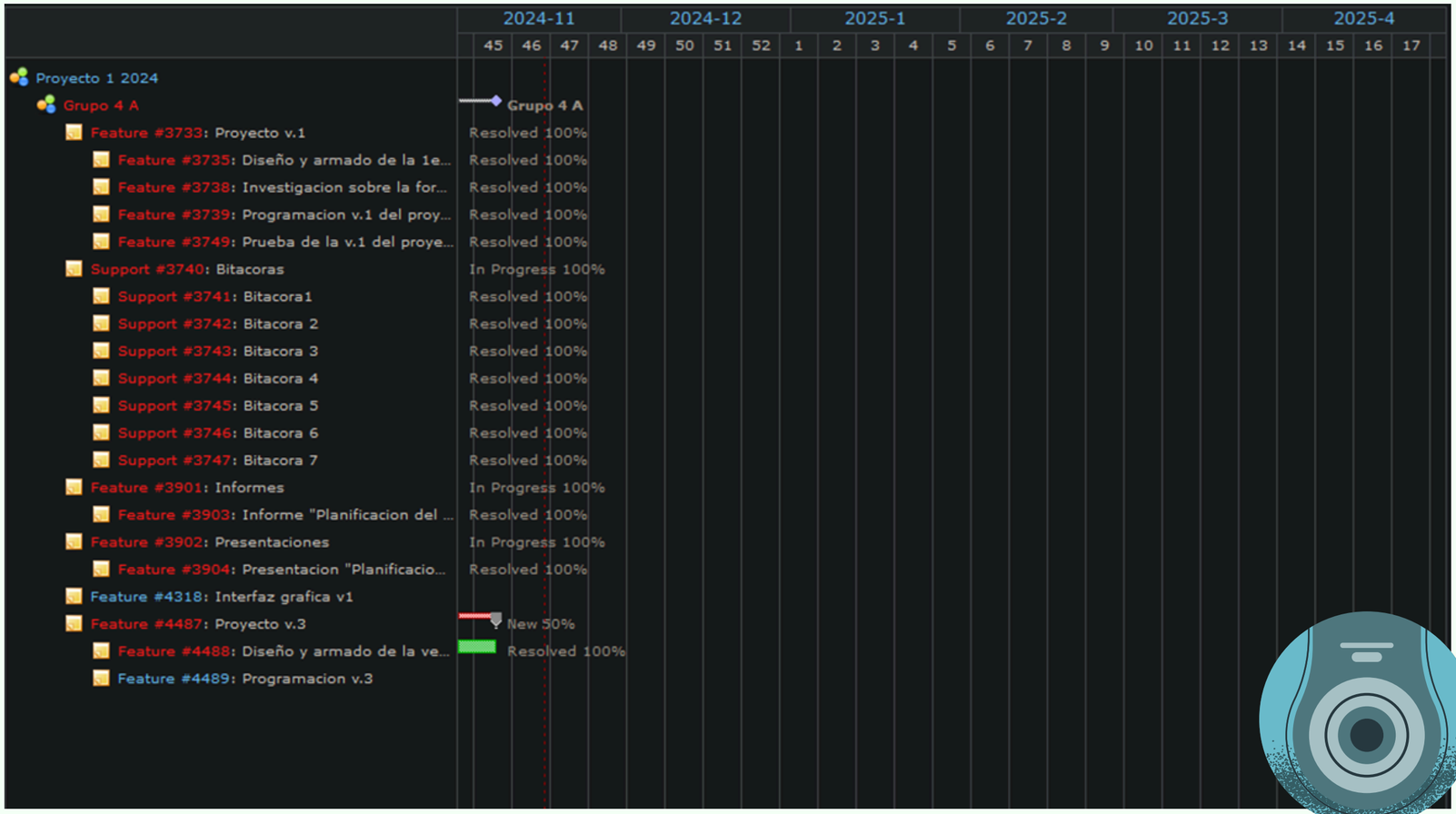
Los medios telemáticos que se utilizarán para la comunicación son Whatsapp para cosas específicas y Discord, siendo Discord el más importante por su fácil uso de creación de grupo, junto con su chat de texto y voz y la facilidad de poder enviar todo tipo de archivos con la cual se puede crear una gran organización de la información.

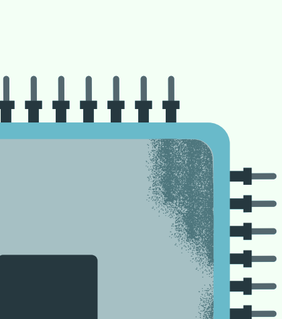
PLANIFICACIÓN DEL PROYECTO

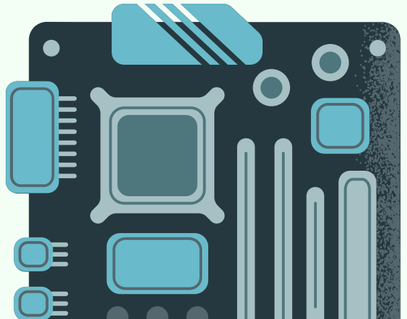
ASIGNACIÓN DE TIEMPO

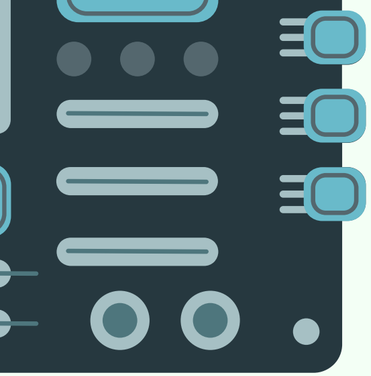
- Planificación de proyecto: 3-4 Semanas
- Planificación de construcción: 1 Semana
- Planificación de interfaz: 1 Semana

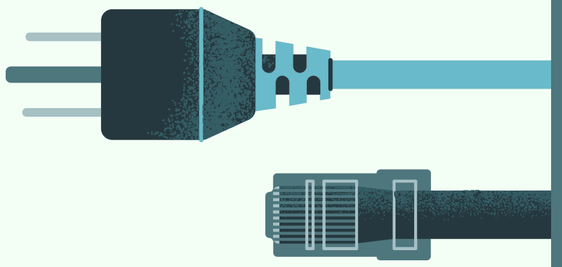
CARTA GANTT









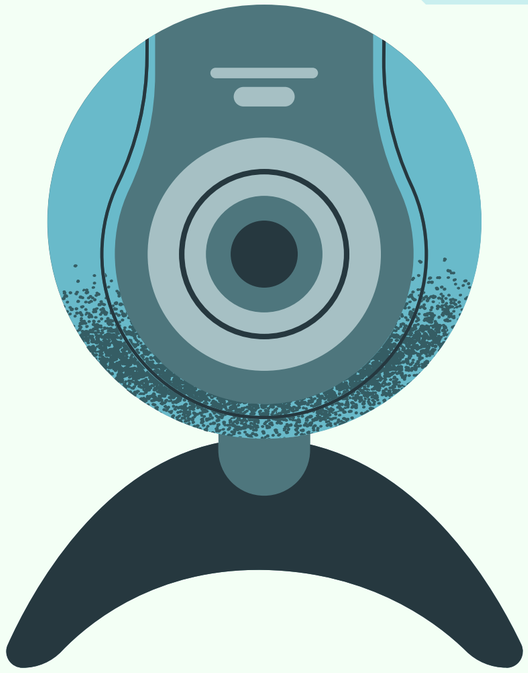


GESTIÓN DE RIESGOS

1. Catastrófico
2. Critico
3. Circunstancial
4. Irrelevante

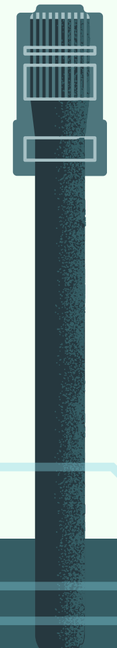
RIESGOS	PROBABILIDAD DE OCURRENCIA	NIVEL DE IMPACTO	POSIBLES SOLUCIONES
FALTA DE PIEZA EN EL ARMADO DEL ROBOT	20%	3	BUSCAR EN BODEGA LA PIEZA FALTANTE, EN CASO DE NO ESTAR USAR UNA PIEZA SIMILAR.
DAÑO EN LA TARJETA SD	10%	1	CAMBIAR LA TARJETA SD POR UNA NUEVA E INSTALAR DE NUEVO EL SISTEMA OPERATIVO.
ERROR EN LA CODIFICACIÓN	60%	2	INVESTIGAR NUEVAMENTE EN SITIOS WEBS OFICIALES, PARA BUSCAR EL POSIBLE ERROR Y COMO SOLUCIONARLO
ENFERMEDAD DE ALGÚN INTEGRANTE DEL EQUIPO.	60%	3	REORGANIZAR EL EQUIPO DE TAL FORMA QUE SE PUEDA CUBRIR EN SU TOTALIDAD LA LABOR ASIGNADA A DICHO MIEMBRO.
FALLO EN EL DISEÑO DEL ROBOT.	30%	2	REALIZAR UN CAMBIO EN EL DISEÑO DEL ROBOT PARA QUE FUNCIONE CORRECTAMENTE.
UNO O MÁS MIEMBROS DEJAN EL PROYECTO.	10%	1	REORGANIZAR LAS TAREAS Y LOS ROLES DE CADA INTEGRANTE.
CATÁSTROFES NATURALES.	10%	1	DEPENDIENDO DEL DAÑO CAUSADO, EL EQUIPO DEBERÍA TRATAR DE REUNIRSE DE MANERA REMOTA O PRESENCIAL.
PÉRDIDA TOTAL DE ARCHIVOS O PROCESOS.	10%	1	RECREAR TODO LO PERDIDO, BASÁNDOSE EN EL CONOCIMIENTO ADQUIRIDO.
QUEDARSE SIN BATERÍA DEL ROBOT.	20%	3	CONECTARLO A UNA FUENTE ELÉCTRICA Y QUE SE RECARGUE.

PLANIFICACIÓN DE LOS RECURSOS



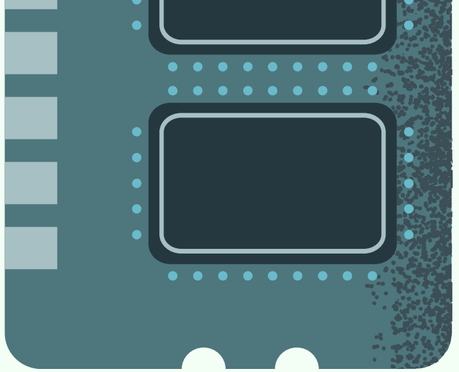
HARDWARE

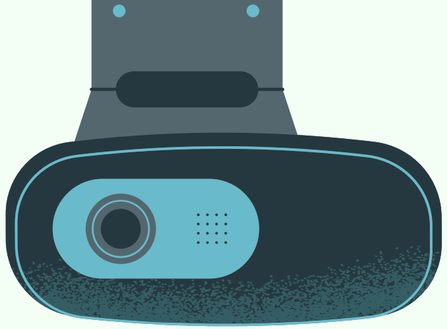
- Tarjeta MicroSD.
- Robot EV3 Mindstorm.
- Wi-fi Dongle.
- Notebook.
- Adaptador MicroSD

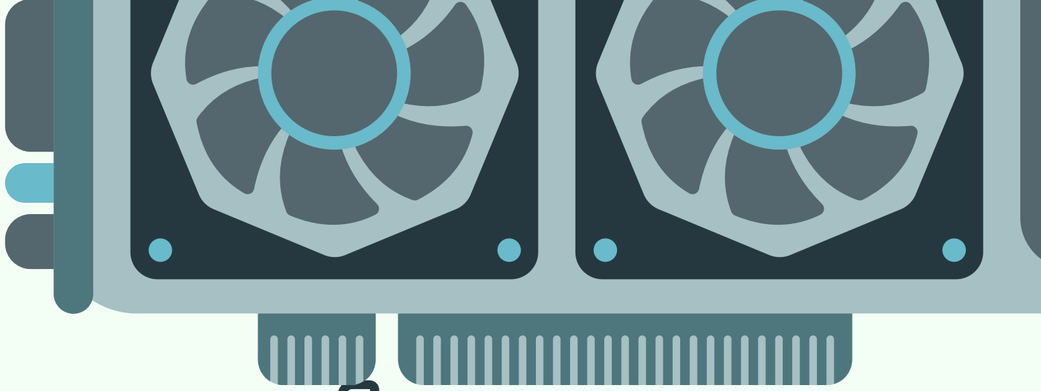


SOFTWARE

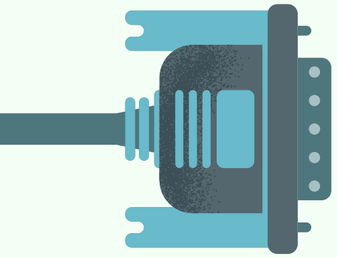
- Visual Studio Code
- Discord
- Ubuntu
- Canva
- Word
- EV3 dev (ev3dev.org)
- WhatsApp
- Python

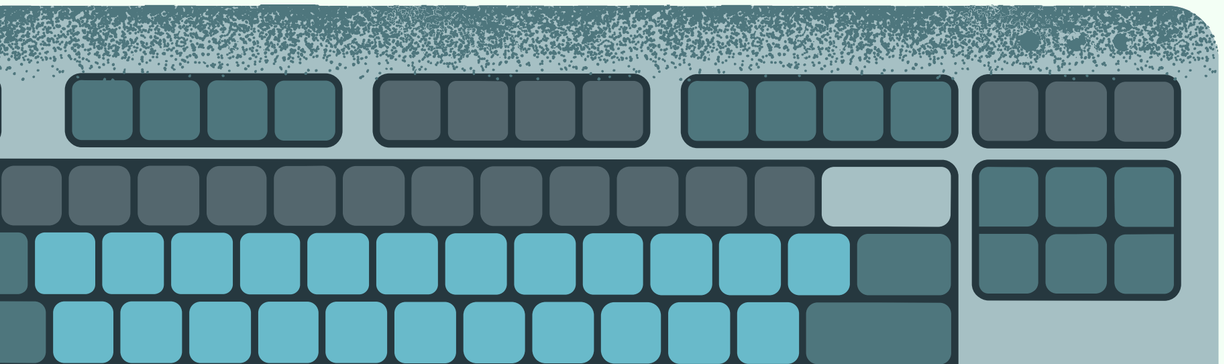


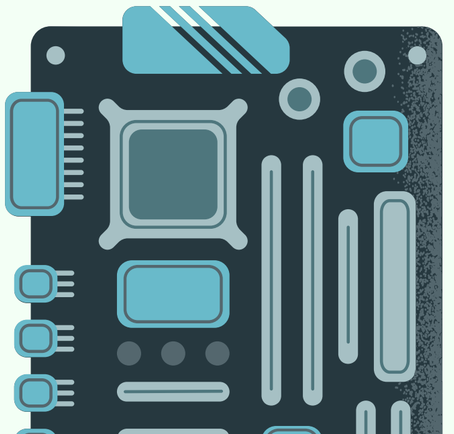




ESTIMACIÓN DE COSTOS





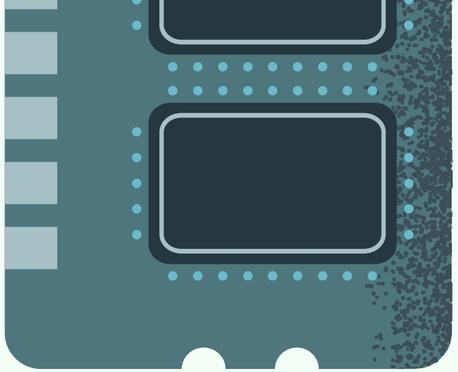


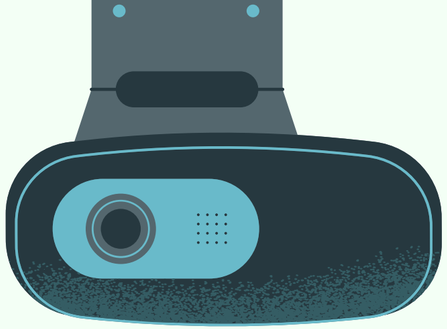
EMPLEADOS

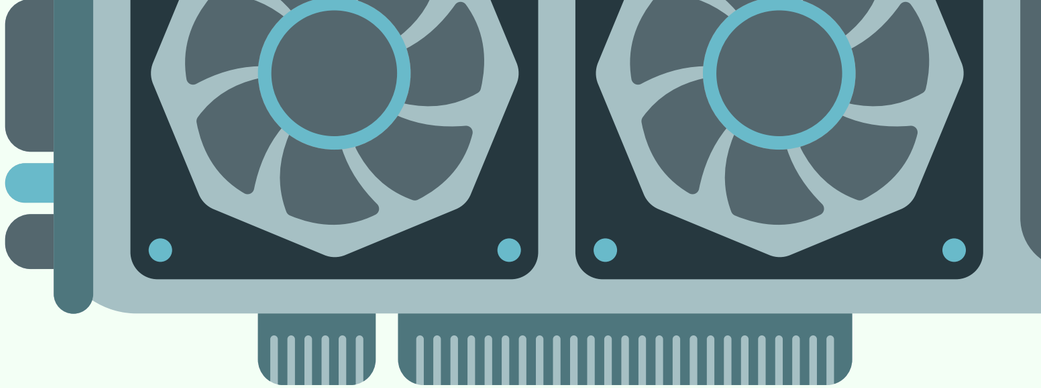
PRODUCTOS	CANTIDADES	PRECIO	CATEGORIA
NOTEBOOK	1 UNIDAD	$700.000	HARDWARE
KIT LEGO MINDSTORMS (EV3)	1 UNIDAD	$1.000.000	HARDWARE
MICRO SD (8 GB)	1 UNIDAD	$5.000	HARDWARE
DONGLE USB WIFI	1 UNIDAD	$7.000	HARDWARE

GENERAL

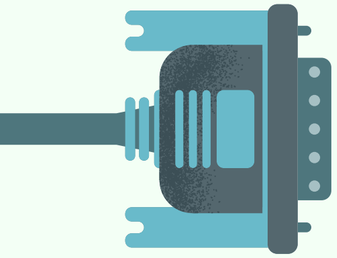
PERSONAL	VALOR HORAS TRABAJADAS	HORAS TRABAJADAS	HORAS TRABAJADAS	HORAS TOTALES TRABAJADAS
ALEX MUÑOZ	34.000	28	4	$ 1.088.000
BENJAMIN FLORES	26.000	28	4	$ 832.000
PATTRICIO MEDINA	32.000	28	4	$ 1.024.000
JONATHAN ORELLANA	32.000	28	4	$ 1.024.000
COSTO TOTAL (MENSUAL)	X	X	X	$3.968.000

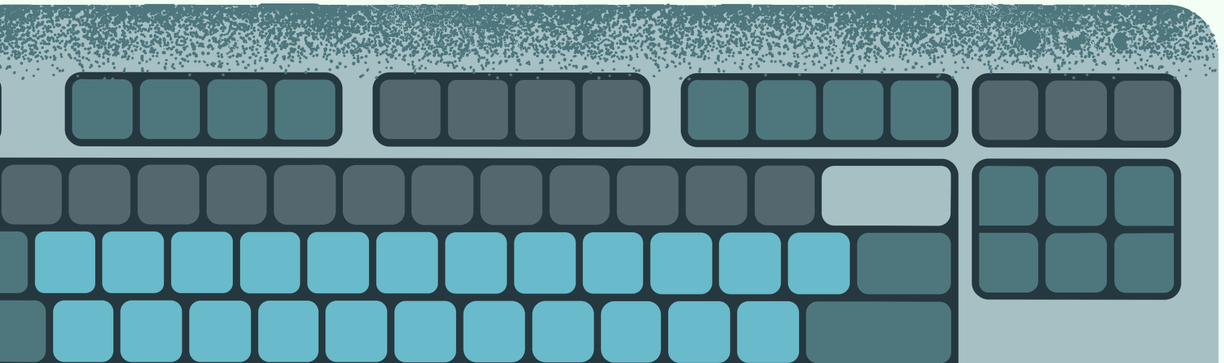


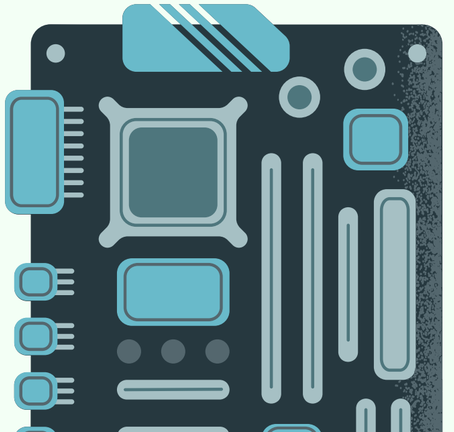




ANÁLISIS Y DISEÑO



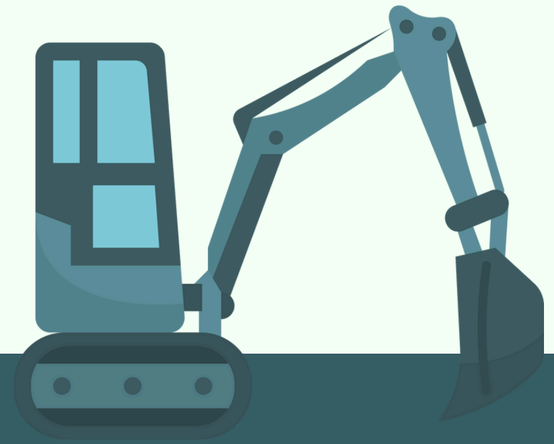




REQUERIMIENTOS

NO FUNCIONALES

- El robot tiene la capacidad de ser controlado con una interfaz gráfica de manera remota
- El robot tiene la capacidad de poder moverse
- El robot tiene la capacidad de mover una garra
- El robot tiene la capacidad de poder agarrar una pelota.



FUNCIONALES

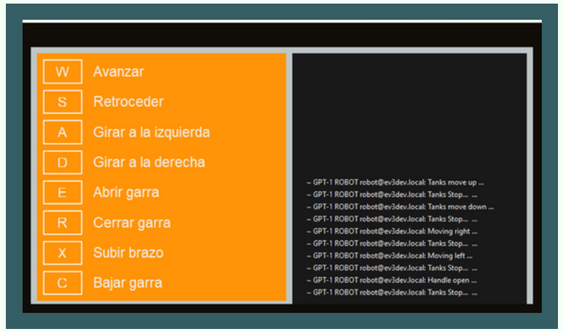
- Cada movimiento del robot se realizará en un tiempo y velocidad estimados.
- La elaboración del código debe ser con el lenguaje de programación Python
- La interfaz gráfica debe tener todos los movimientos del robot tales como moverse hacia adelante, hacías atrás y hacia los lados, además de mover la garra.
- El robot debe ser construido con las piezas de lego kit Mindstorm EV3 Education.

ARQUITECTURA

CLIENTE

CONEXION WIFI

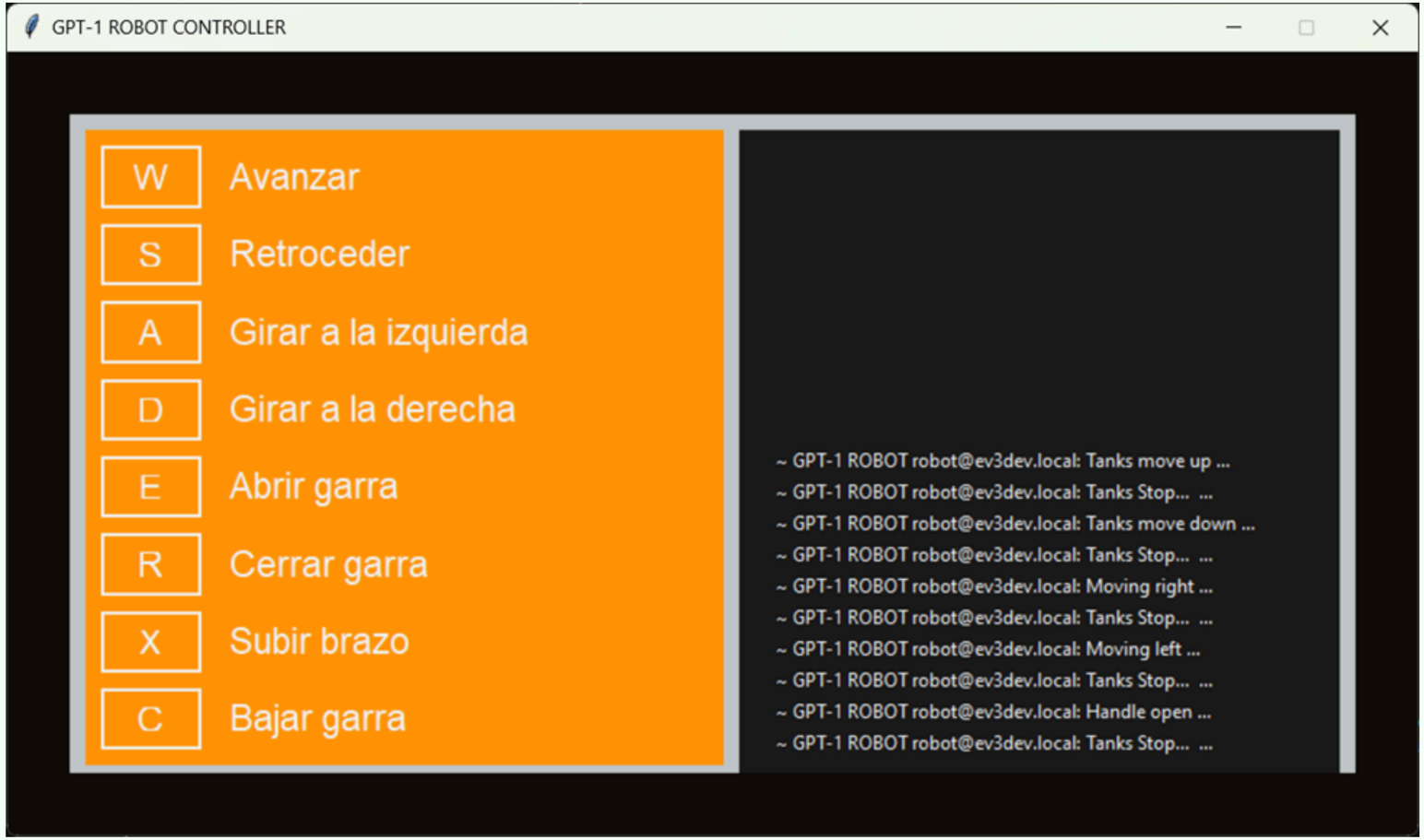
SERVIDOR

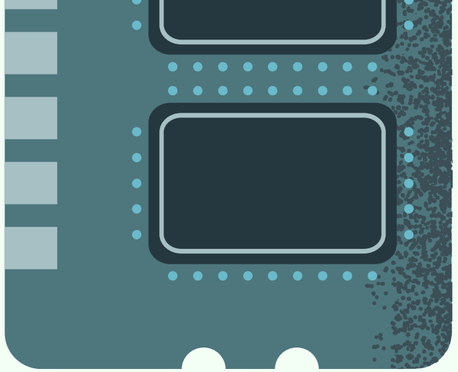


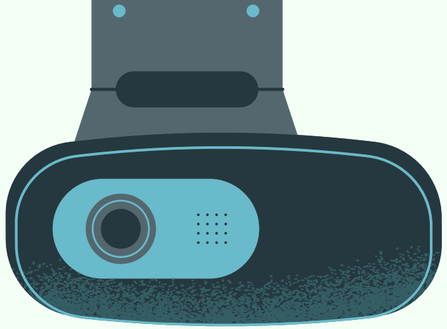
INTERFAZ

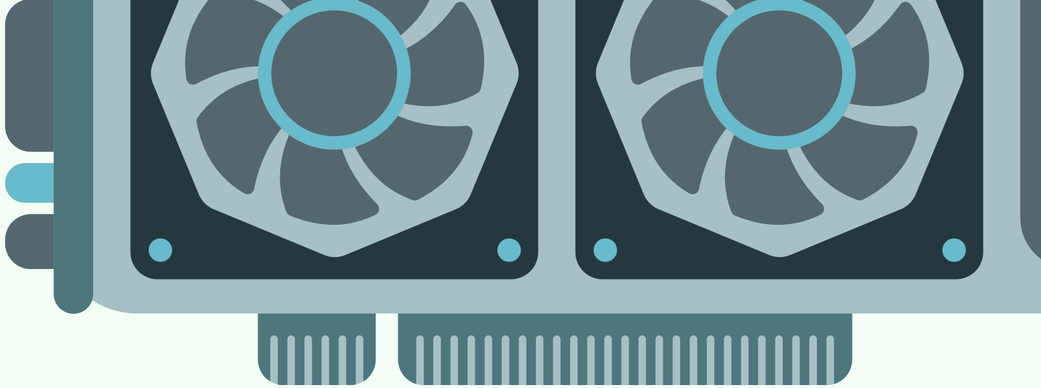
ROBOT

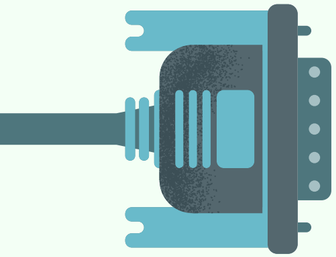
INTERFAZ GRÁFICA



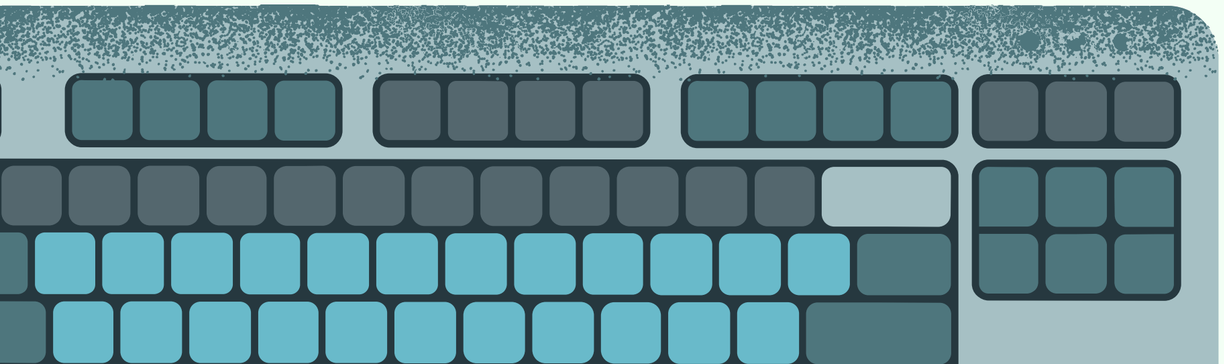


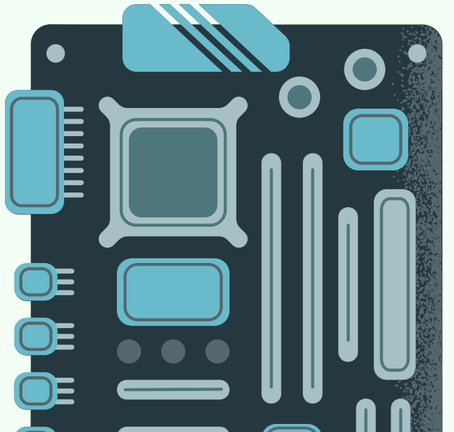






IMPLEMENTACION





FUNDAMENTOS DE CARGA

Cinemática del movimiento
Desplazamiento lineal
- Posición: Ubicación de la carga o robot
- Velocidad: Rapidez del robot o de la carga
- Aceleración: Cambio de la velocidad del movimiento

Formulas:

$$d = v \cdot t + \frac{1}{2} a \cdot t^2$$

- d: Desplazamiento lineal
- v: Velocidad inicial
- t: Tiempo
- a: Aceleración lineal

Dinámica y fuerzas involucradas
Fuerzas principales:
- Fuerza de tracción: Generada por motores que impulsan el movimiento
- Fricción: Entre las ruedas y suelo que afecta el desplazamiento
- Fuerzas inerciales: Cuando el robot cambie de velocidad estas fuerzas tienden a resistir el movimiento.

Formulas:

$$F = m \cdot a$$

- m: Masa de la carga
- a: Aceleración

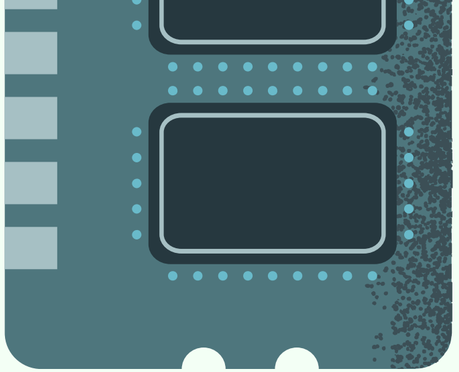
Centro de gravedad
La posición del centro de gravedad del robot y la carga es crucial ya que evita que el robot llegue a volcarse.

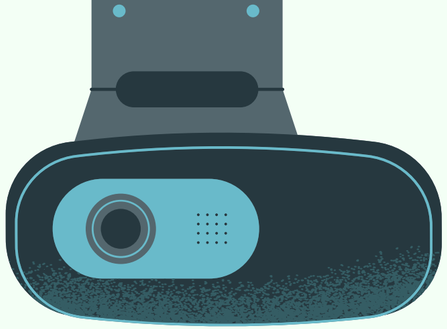
$$g = 9{,}8 m/s^2$$

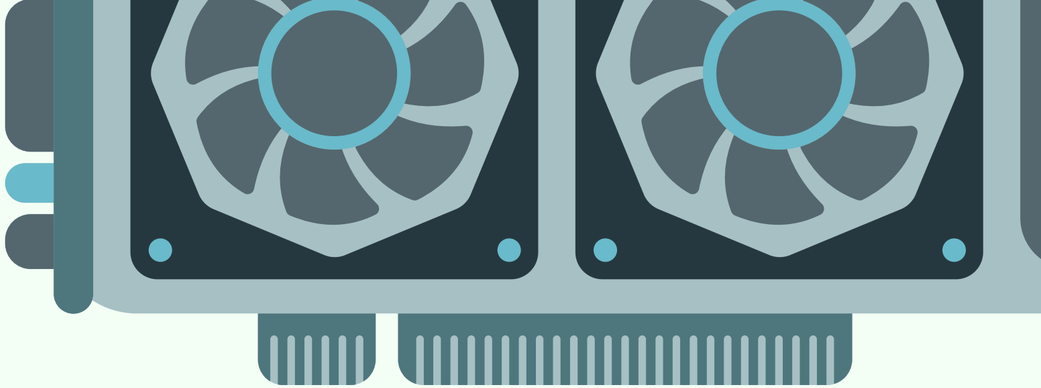
Energía y trabajo
Levantar una carga implica realizar trabajo contra la fuerza gravitatoria.
Formula:

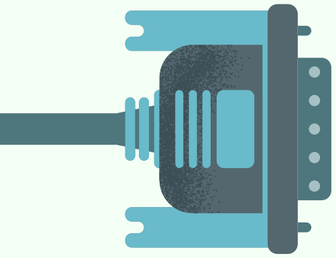
$$W = F \cdot d \cdot \cos(\theta)$$

- W: Trabajo realizado
- F: Fuerza aplicada
- d: Distancia recorrida de la carga
- θ: Angulo entre la fuerza y dirección del movimiento

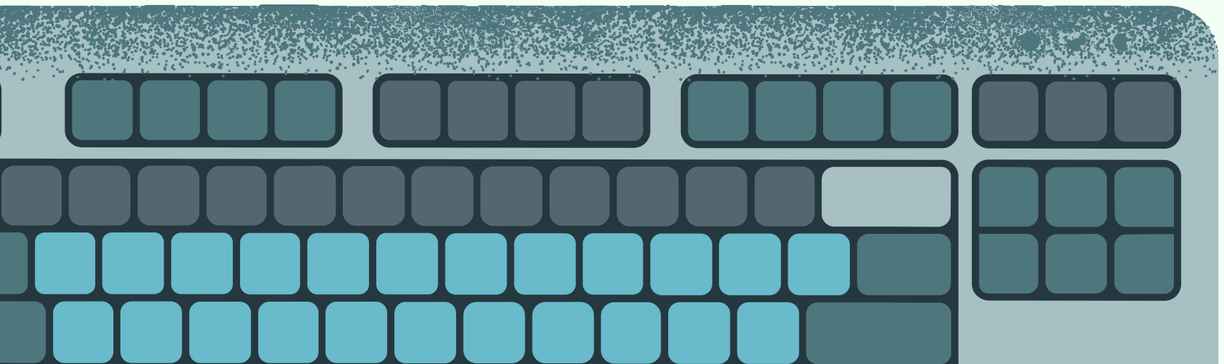


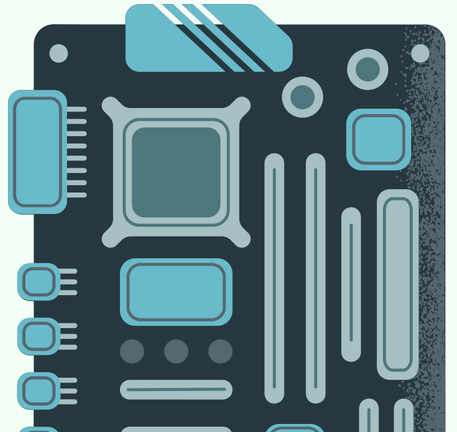




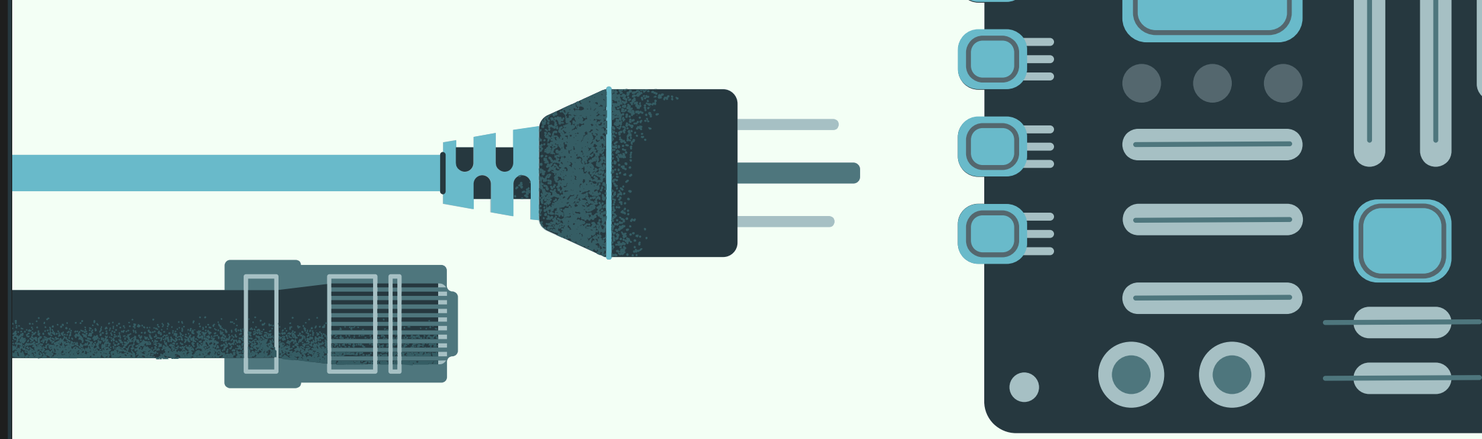


DESCRIPCIÓN DE LOS PROGRAMAS



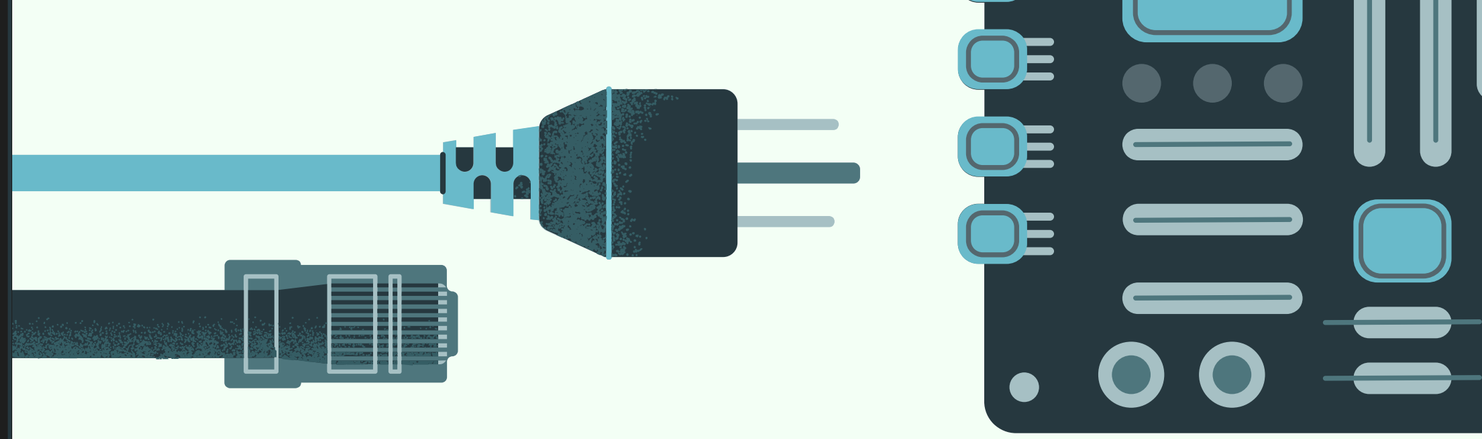


```python
import socket
from library2 import *

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()

speed = 100

while True:
    rawByte = connect.recv(1)
    char = rawByte.decode('utf-8')

    if (char == 'w'):
        moveUp(speed)

    if (char == 's'):
        moveDown(speed)

    if (char == 'd'):
        moveRight(speed)

    if (char == 'a'):
        moveLeft(speed)

    if (char == 'e'):
        handle_open()

    if (char == 'r'):
        handle_close()

    if (char == 'x'):
        handleUp()

    if (char == 'c'):
        handleDown()

    if (char == 'z'):
        stopHandle()

    if (char == ' '):
        stop()

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



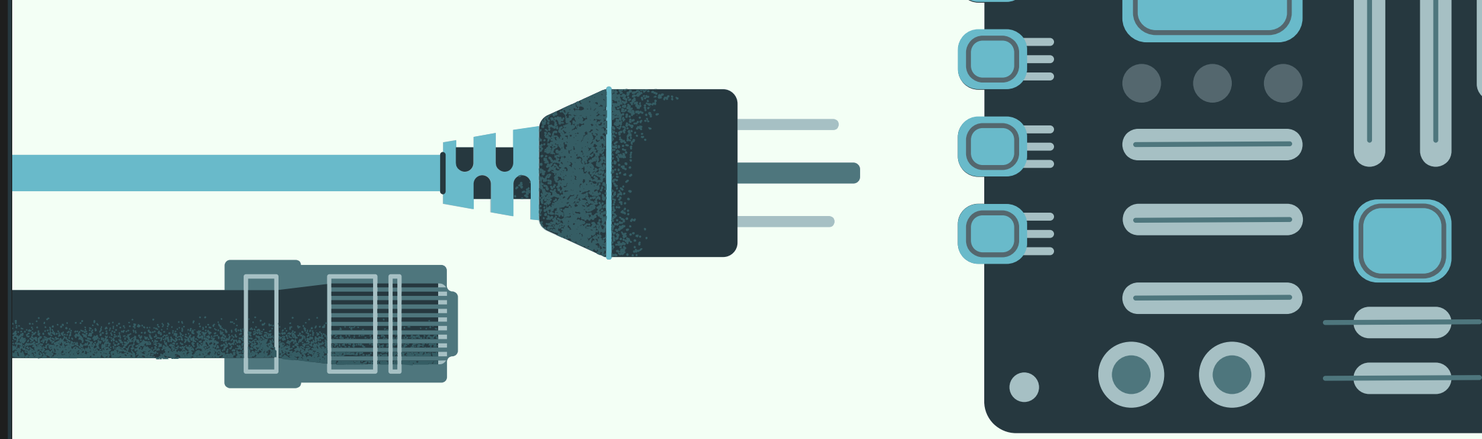
SERVER

```python
import time
from ev3dev2.motor import MoveTank , OUTPUT_A, OUTPUT_B, LargeMotor , OUTPUT_C, MediumMotor, OUTPUT_D

mov_garra = LargeMotor(OUTPUT_C)
garra = MediumMotor(OUTPUT_D)
tankmoves = MoveTank(OUTPUT_A, OUTPUT_B)

def moveUp(speed):
    # Avanza el robot con velocidad variable
    print("Velocidad adeladnte")
    tankmoves.on(speed, speed)

def moveDown(speed):
    # Retrocede el robot con velocidad variable
    print("Velocidad atras")
    tankmoves.on(-speed, -speed)

def moveLeft(speed):
    # Rota a la Izquierda el robot con velocidad variable
    print("Rotando izquierda")
    tankmoves.on(-speed // 5, speed // 5)

def moveRight(speed):
    # Rota a la Derecha el robot con velocidad variable
    print("Girarando derecha")
    tankmoves.on(speed // 5, -speed // 5)

def handleUp():
    # Sube el brazo del robot
    print("Subiendo brazo")
    mov_garra.on(-7)
    time.sleep(4)
    mov_garra.off()

def handleDown():
    # Baja el brazo del robot
    print("Bajando brazo")
    mov_garra.on(7)
    time.sleep(4)
    mov_garra.off()

def handle_open():
    # Abre la garra del robot
    print("Abriendo")
    garra.on(3)

def handle_close():
    # Cierra la garra del robot
    print("Cerrando")
    garra.on(-3)

def stopHandle():
    # Detiene em movimiento de laga garra
    print("Deteniendo garra")
    garra.off()

def stop():
    # Detiene el moviento del robot
    print("Detenido")
    tankmoves.off()
```



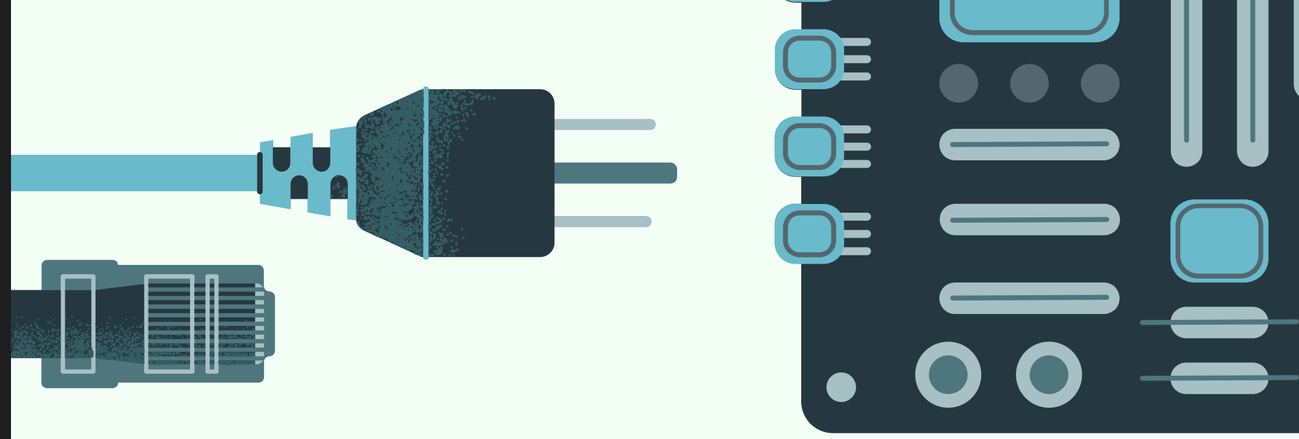
LIBRARY

```python
import tkinter as tk

from library2 import *

class FrameFondo(tk.Frame):
    def __init__(self, master, color, **kwargs):
        super().__init__(master, **kwargs)
        self.configure(bg=color)
        self.pack(pady=40, padx=40, fill="both", expand=True)
        self.columnconfigure(0, weight=1)
        self.columnconfigure(1, weight=3)
        self.rowconfigure(0, weight=1)
        self.rowconfigure(1, weight=2)

class FrameUno(tk.Frame):
    def __init__(self, master, color, **kwargs):
        super().__init__(master, **kwargs)
        self.configure(bg=color)
        self.grid(row=0, column=0, sticky='nwse', pady=(10, 5), padx=10)

        self.configure(height=100)

        self.columnconfigure(0, weight=1)
        self.columnconfigure(1, weight=10)
        self.rowconfigure((0, 1, 2, 3, 4, 5, 6, 7, 8), weight=1)

        self.crearFrameTecla('W', (10, 5))
        self.crearFrameTecla('S', 5)
        self.crearFrameTecla('A', 5)
        self.crearFrameTecla('D', 5)
        self.crearFrameTecla('E', 5)
        self.crearFrameTecla('R', 5)
        self.crearFrameTecla('X', 5)
        self.crearFrameTecla('C', 5)
        self.crearFrameTecla('Z', (5, 10))

        self.descripcion('Avanzar', 0, (10, 5))
        self.descripcion('Retroceder', 1, 5)
        self.descripcion('Girar a la izquierda', 2, 5)
        self.descripcion('Girar a la derecha', 3, 5)
        self.descripcion('Abrir garra', 4, 5)
        self.descripcion('Cerrar garra', 5, 5)
        self.descripcion('Subir brazo', 6, 5)
        self.descripcion('Bajar brazo', 7, 5)
        self.descripcion('Detener brazo', 8, (5, 10))

    def crearFrameTecla(self, texto, pady):
        frame = tk.Frame(self, bg='#FF9408', bd=2, relief="solid")
        frame.grid(sticky='nwse', pady=pady, padx=10)

        label = tk.Label(frame, text=texto, fg='#F3F4F5', bg='#FF9408')
        label.pack(pady=10, padx=10)

        frame.columnconfigure(0, weight=1)
        frame.rowconfigure(0, weight=1)

    def descripcion(self, texto, fila, pady):
        label = tk.Label(self, text=texto, fg='#F3F4F5', anchor='w', font=('Helvetica', 17), bg=self.cget('bg'))
        label.grid(row=fila, column=1, pady=pady, padx=5, sticky='w')
```



INTERFAZ

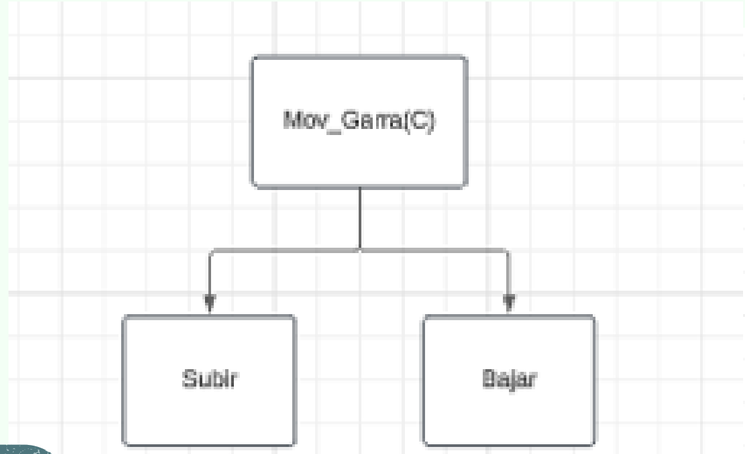
```python
class FrameDos(tk.Frame):
    def __init__(self, master, color, **kwargs):
        super().__init__(master, **kwargs)
        self.configure(bg=color)
        self.grid(row=0, column=1, sticky='nwse', rowspan=2, pady=(10), padx=(0, 10))

        self.memoria = ''
        self.contador = 0
        self.labels = []

    def agregar_registro(self, tecla):
        if tecla.upper() in ('W', 'S', 'A', 'D', 'E', 'R', 'SPACE') and tecla.upper() != self.memoria:
            self.memoria = tecla.upper()
            print("Tecla presionada: {0}".format(self.memoria))

            texto = ''
            opcion = self.memoria
            velocidad = 100

            if self.memoria == 'SPACE':
                texto = 'Tanks Stop.... '
                stop()

            if self.memoria == 'W':
                texto = 'Tanks Move Up....'
                moveUp(velocidad)

            if self.memoria == 'S':
                texto = 'Tanks Move Down.... '
                moveDown(velocidad)

            if self.memoria == 'A':
                texto = 'Moving Left.... '
                moveLeft(velocidad)

            if self.memoria == 'D':
                texto = 'Moving right.... '
                moveRight(velocidad)

            if self.memoria == 'E':
                texto = 'Handle Open.... '
                handle_open()

            if self.memoria == 'R':
                texto = 'Handle Close.... '
                handle_close()

            if self.memoria == 'X':
                texto = 'Handle Up.... '
                handleUp()
            if self.memoria == 'C':
                texto = 'Handle Down.... '
                handleDown()

            if self.memoria == 'Z':
                texto = 'Handle Stop .... '
                stopHandle()

            registro = tk.Label(self, text="> GPT-1 ROBOT robot@ev3dev.local :  {0} ...".format(texto), bg=self.cget('bg'), fg='#F3F4F5')
            self.labels.append(registro)

            base_y = 380 - (len(self.labels) * 20)

            for i, label in enumerate(self.labels):
                label.place(x=20, y=base_y + (i * 20))

class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title("GPT-1 ROBOT CONTROLLER")
        self.geometry("900x500")
        self.resizable(width=0, height=0)
        self.configure(bg="#100C08")

        frame_fondo = FrameFondo(self, '#C0C7C8', relief="solid", bd=2)
        frame_uno = FrameUno(frame_fondo, '#FF9408', relief="solid", bd=2)
        self.frame_d0s = FrameDos(frame_fondo, '#191919', relief="solid", bd=2)

        self.bind("<Key>", self.tecla_pulsada)

    def tecla_pulsada(self, event):
        self.frame_d0s.agregar_registro(event.keysym)

if __name__ == "__main__":
    app = App()
    app.mainloop()
```

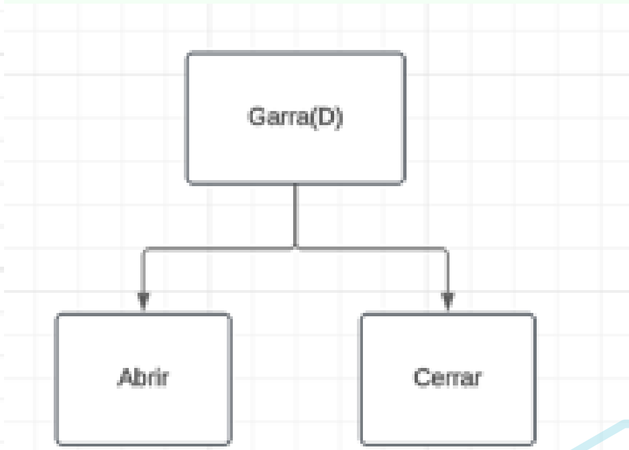


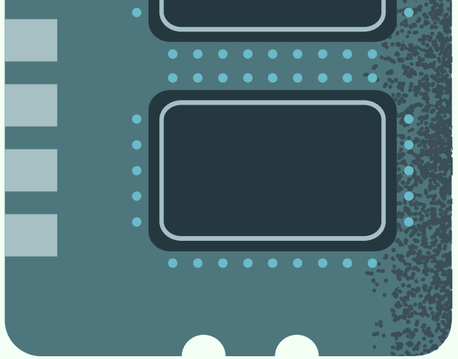
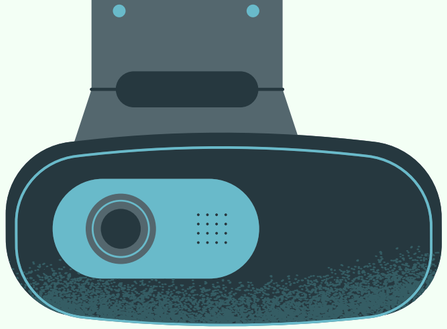
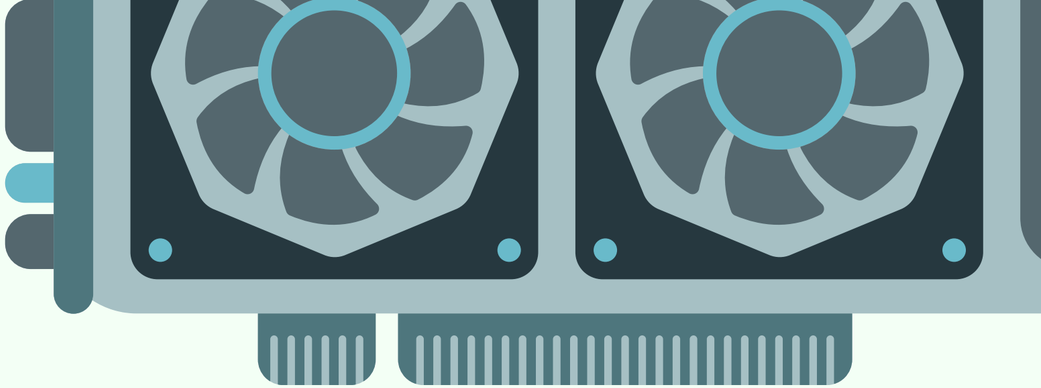
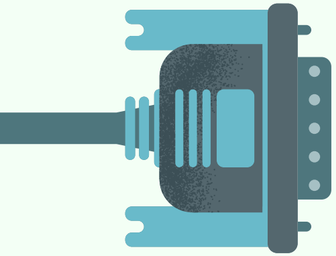
INTERFAZ

DIAGRAMAS

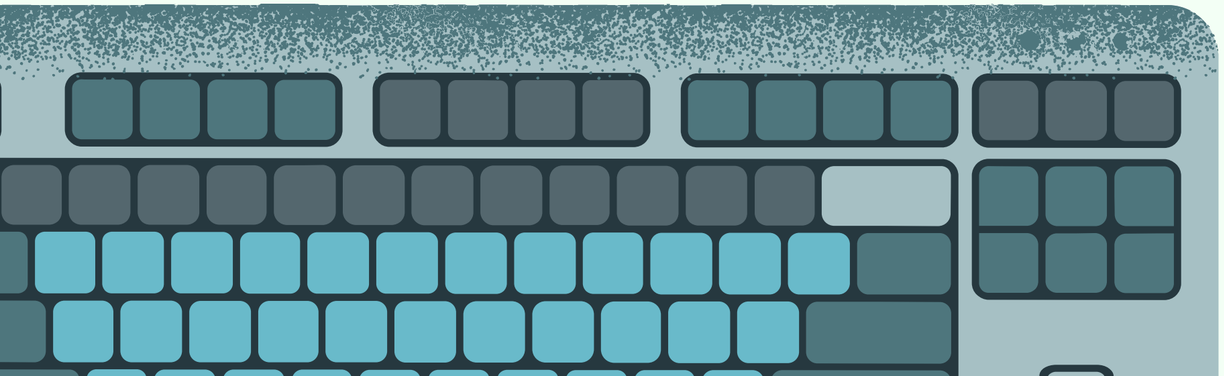

Tankmoves(A,B)
Derecha
Izquierda
Atras
Adelante

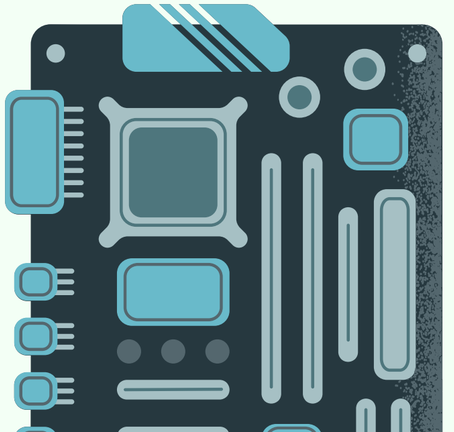

Mov_Garra(C)
Subir
Bajar


Garra(D)
Abrir
Cerrar

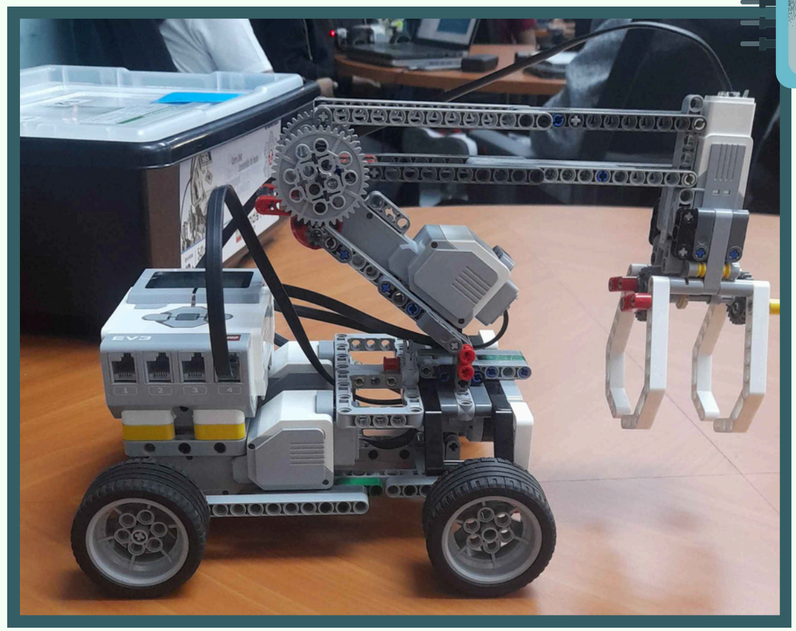
RESULTADO

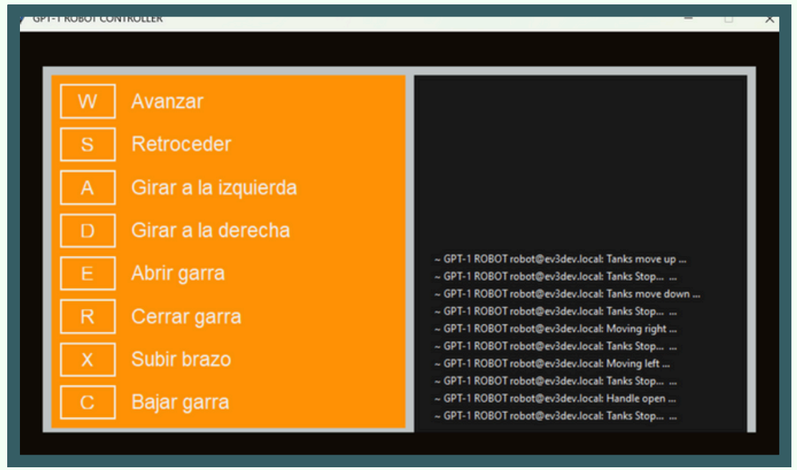



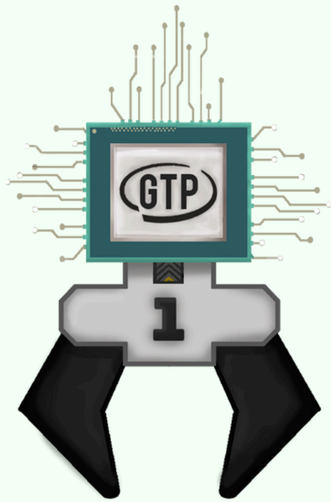


ESTADO ACTUAL

- El robot ya se encuentra construido en su totalidad.
- Cuenta con sus funciones ya definidas tanto como su dirección, movimiento del brazo al igual que de la garra
- Control del robot desde una interfaz gráfica mediante Tkinter y la creación de su servidor
- Wiki del proyecto Actualizada
- Carta Gantt actualizadas



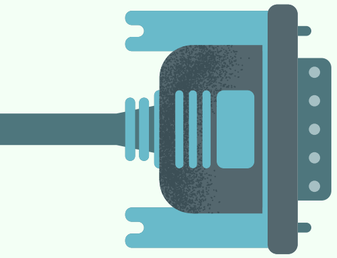


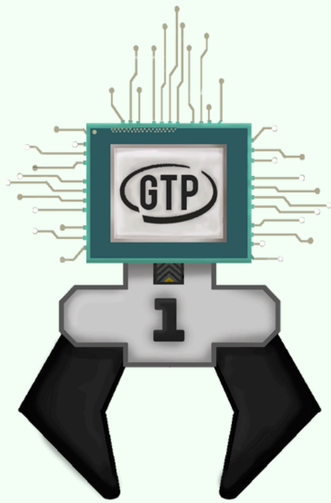


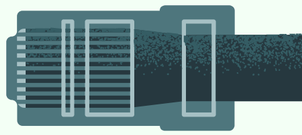
PROBLEMAS ENCONTRADOS Y SOLUCIÓN PROPUESTA

PROBLEMAS ENCONTRADOS	SOLUCIONES
ERROR DE COMPATIBILIDAD POR VERSIONES SE UTILIZARON FUNCIONES DE PYTHON NO COMPATIBLES CON LA VERSIÓN INSTALADA EN EL ROBOT EV3DEV (SE USÓ LA VERSIÓN 3.13, MIENTRAS QUE EL ROBOT TIENE LA 3.5.2).	FUE NECESARIO REALIZAR VARIAS CORRECCIONES AL CÓDIGO PARA ASEGURAR SU COMPATIBILIDAD. ALGUNOS DE LOS CAMBIOS REALIZADOS INCLUYEN: • MODIFICACIÓN DE LA INTERFAZ: CAMBIO DE LA BIBLIOTECA CUSTOMTKINTER A TKINTER. • AJUSTES EN LAS FUNCIONES DE LIBRARY2.PY. • SUSTITUCIÓN DEL CONDICIONAL MATCH-CASE POR EL CONDICIONAL IF. • CAMBIO DE PRINT CON FORMATO (F-STRING) A PRINT ESTÁNDAR.
A LA HORA DE CREAR LA VERSIÓN 2 DEL ROBOT TUVIMOS UN PROBLEMA POR LA CANTIDAD DE PIEZAS QUE LLEGABA A USAR EL ROBOT Y UN TAMAÑO POR LO QUE SE DESCARTÓ DIRECTAMENTE SU CREACIÓN)	CREAR UN DISEÑO NUEVO QUE SEA MÁS SIMPLE Y EFICIENTE PARA QUE EL ROBOT NO TENGA UN TAMAÑO EXAGERADO Y EL GASTO DE PIEZAS NO SEAN TAN ALTAS, CON LO CUAL CREAMOS LA VERSIÓN 3 DEL ROBOT SIENDO LA ACTUAL Y MÁS EFICIENTE A LAS ANTERIORES

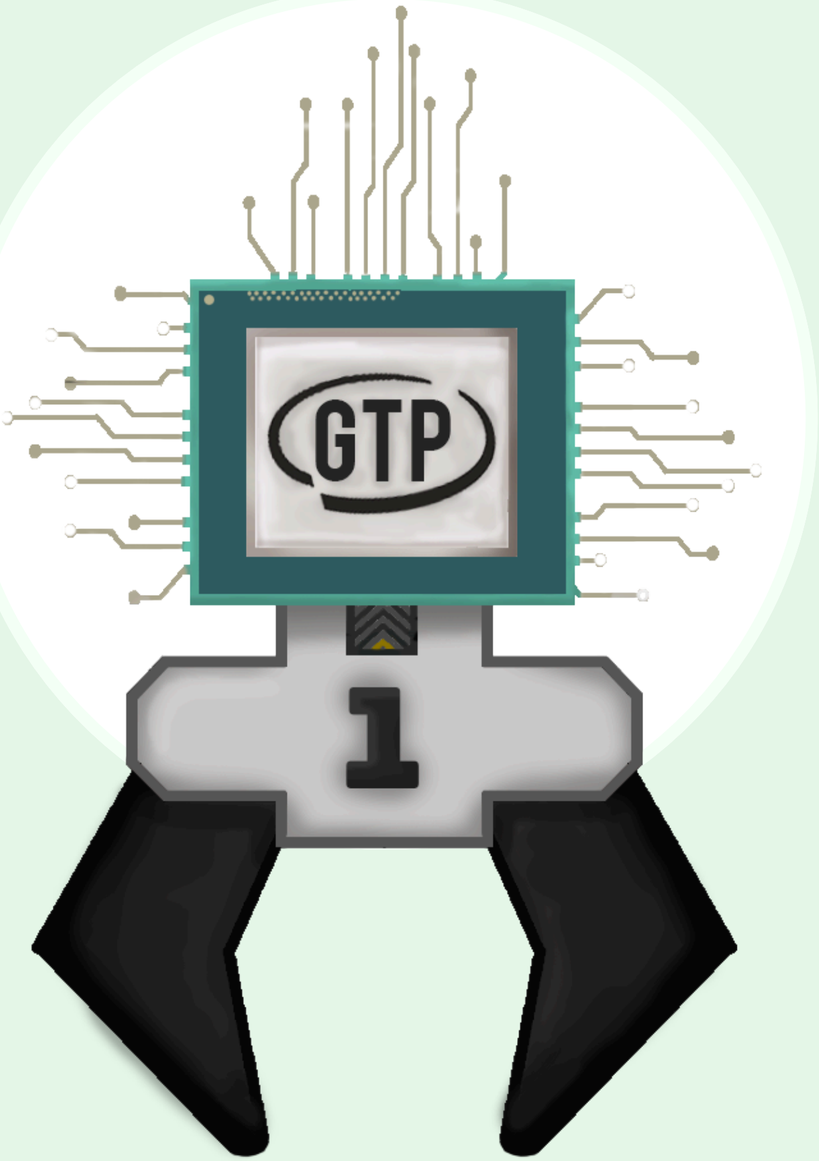
CONCLUSIÓN







GTP-1



Integrantes:
Benjamin Flores
Alex Muñoz
Jonathan Orellana
Pattricio Medina

UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

Ingeniería
Computación e Informática