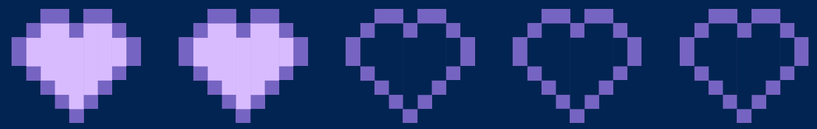
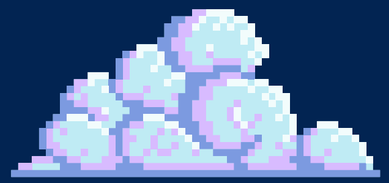
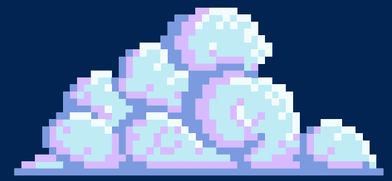
UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

Ingenierí@
Computación e Informática





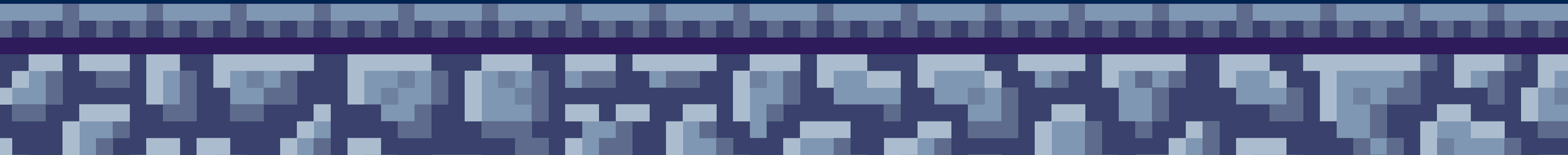


ALLIGATOR 3000

PROYECTO I

PROFESOR: HUMBERTO URRUTIA

START



01

ANÁLISIS Y DISEÑO

REQUERIMIENTOS

NO FUNCIONALES

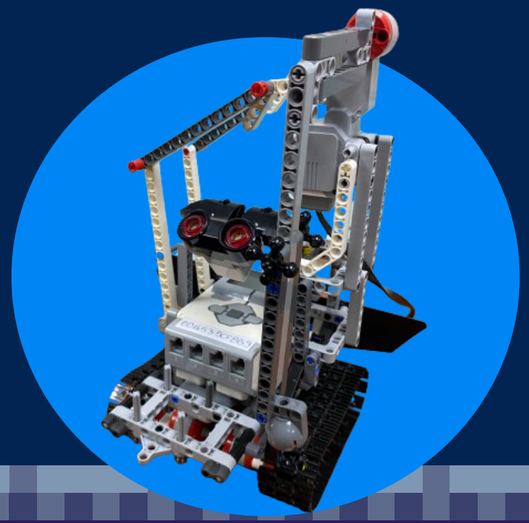
- La interfaz gráfica contendrá los movimientos necesarios para movilizar el robot. Además de ser fácil de usar.
- El robot debe ser construido con piezas del kit Lego Mindstorm EV3 Education.
- La estructura del robot debe ser estable .
- El robot debe tener la habilidad de impactar una pelota haciendo uso de formulas físicas.

FUNCIONALES

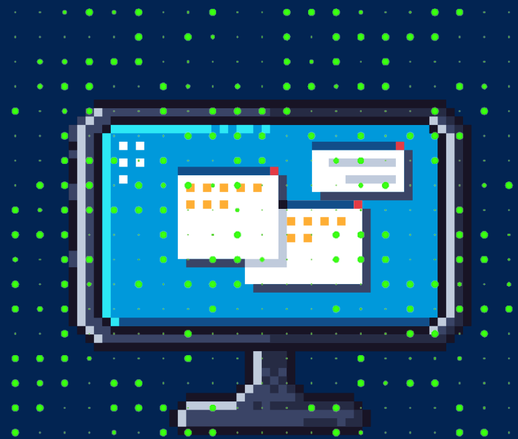
- El robot debe tener la capacidad de moverse.
- Los usuarios deben controlar el robot a través de una interfaz gráfica en un dispositivo técnico.
- El robot debe poder golpear una pelota.

ARQUITECTURA

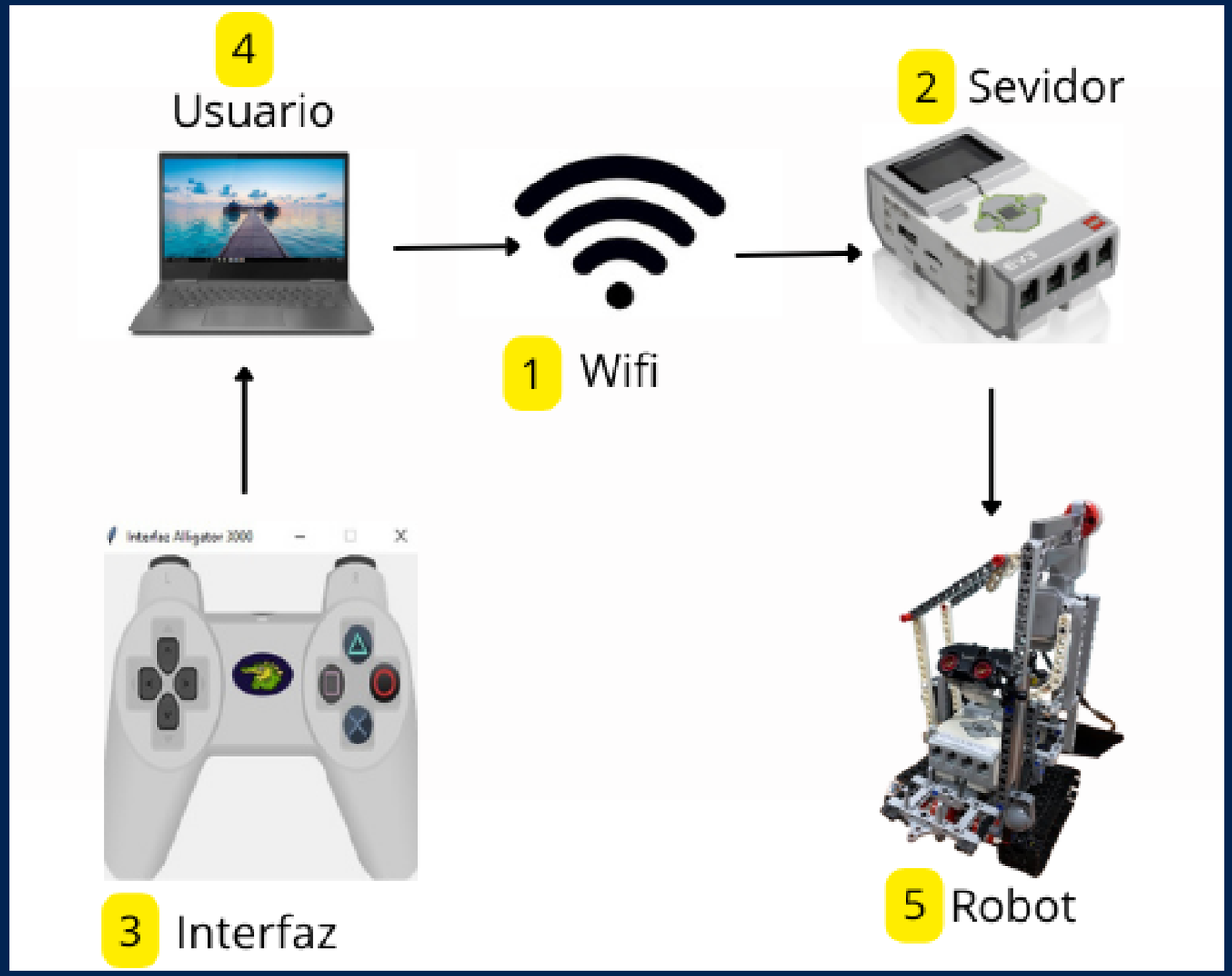
Cuerpo del robot.



Programación



CONEXIÓN DEL ROBOT



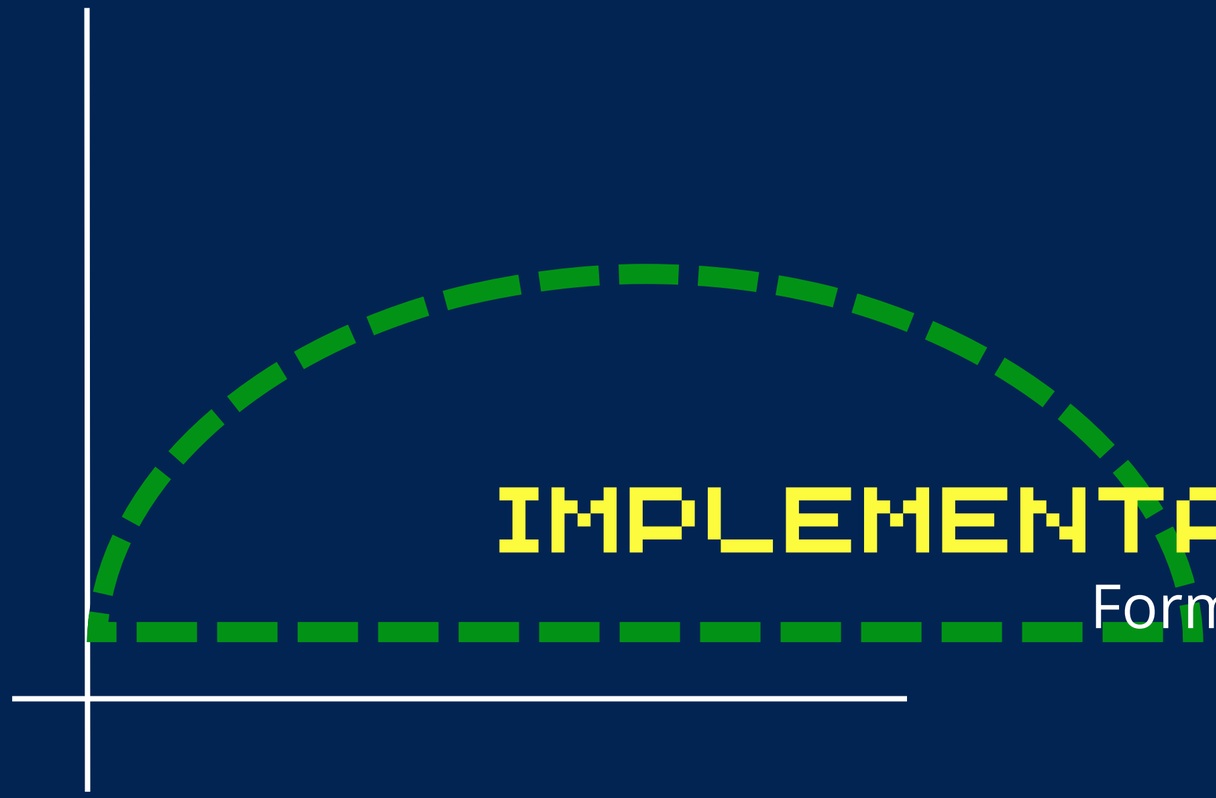
INTERFAZ GRÁFICA

1. Flechas direccionales para indicar las distintas orientaciones del robot.

2. "Golpear", botón que contiene un círculo rojo, el cual activará la función de golpear la pelota.

3. "Conectar", botón que contiene la “X” que establece la conexión entre el cliente y el servidor en el EV3

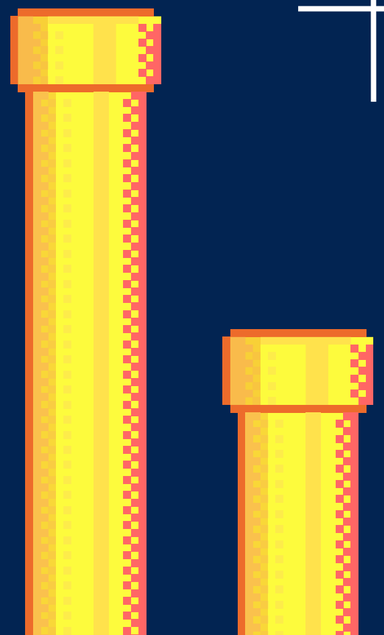
4. Para finalizar la conexión, se utiliza el botón que contiene un triángulo verde.

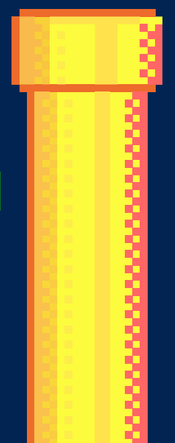


02

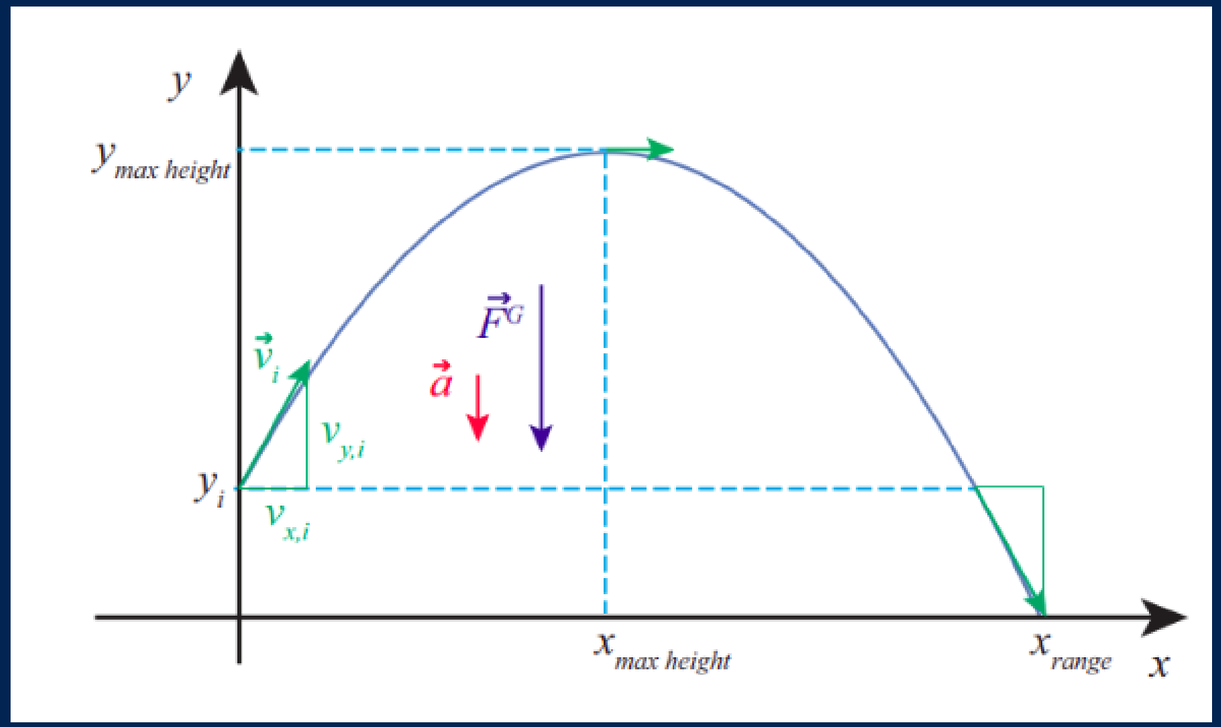
IMPLEMENTACIÓN

Formulas físicas





FUNDAMENTOS DEL MOVIMIENTO PARABÓLICO.



DATOS:

g =9,8ms^2

∅ =30°

Yo = 0,9 m

Yf = 0 m

Xf = 0,39 m

El tiempo del lanzamiento del proyectil:

$Xf = Xo + Vo \cos \theta \, t$

$0,39 = 0 + Vo \cos 30^\circ \, t$

$Vo = 0,39 / \cos 30^\circ * t$

$Yf = Yo + Vo \operatorname{sen} \theta \, t + 1/2 * (-g) \, t^2$

$0 = 0,9 + Vo \operatorname{sen} 30^\circ \, t - 4,9 \, t^2$

$0 = 0,9 + (0,39 / \cos 30^\circ \, t) * (\operatorname{sen} 30^\circ \, t) - 4,9 \, t^2$

$0 = 0,9 + 0,39 \tan 30^\circ - 4,9 \, t^2$

$4,9 \, t^2 = 1,1251$

$t^2 = 0,2296$

$t = \sqrt{0,2296} = 0,4791 \, (s)$

El eje Y :

$Yf = Yo + Voy \, t + 12(-g) \, t^2$

$0 = 0,9 + 0,4791 \, Voy - 4,9 \, (0,4791)^2$

$0,2247 = 0,4791 \, Voy$

$Voy = 0,4690 \, (m/s)$

El eje X :

$Xf = Xo + Vox \, t$

$0,39 = 0 + 0,4690 \, Vox$

$Vox = 0,39 / 0,4690$

$Vox = 0,93 \, (m/s)$

Por lo tanto, al obtener las velocidades en el plano X e Y, podemos determinar la velocidad inicial del proyectil:

$Vo = 0,39 \cos 30^\circ * 0,460$

$Vo = 0,93 \, (m/s)$

03

CÓDIGO

server.py

server.py * ×

```python
#!/usr/bin/env python3
import socket
from Function 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 == 'p'):
        Brazo()
    if (char == ' '):
        print("Alto")
    if (char == 'q'):
```

Function.py

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

sound = Sound()

ruedas = MoveTank(OUTPUT_B,OUTPUT_C)
brazo= LargeMotor(OUTPUT_D)

def Adelante():
    sound.speak('Chao Polette')
    ruedas.on(SpeedPercent(50),SpeedPercent(50))
    time.sleep(1)
    ruedas.off()

def Atras():
    ruedas.on(SpeedPercent(-70),SpeedPercent(-70))
    time.sleep(1)
    ruedas.off()

def Izquierda():
    ruedas.on(SpeedPercent(70),SpeedPercent(-70))
    time.sleep(1)
    ruedas.off()

def Derecha():
    ruedas.on(SpeedPercent(-70),SpeedPercent(70))
    time.sleep(1)
    ruedas.off()
```

```python
def Brazo():
    brazo.on_for_rotations(30,1)
    time.sleep(1)
    ruedas.off()
```

Interfaz

```python
#!/usr/bin/env python3
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
import socket
import time
import sys
from PIL import Image, ImageTk

#Modificar imagenes para la apariencia de la ventana y botones

def resize_image(image_path, new_width, new_height):
    original_image = Image.open(image_path)
    resized_image = original_image.resize((new_width, new_height), Image.ANTIALIAS)
    return ImageTk.PhotoImage(resized_image)

#Función para abrir una ventana y solicitar la IP

def getAddres():
    w_ip = Tk()
    w_ip.geometry("300x100")

    dato = StringVar(w_ip)
    w_ip.title("Configurar Ip")
    mensaje= Label(w_ip, text="Ingrese la dirección IP para efectuar la\n conexión con el servidor").place(x=40,y=0)
    ip = ttk.Entry(w_ip,textvariable=dato).place(x=40,y=40)
    button = Button(w_ip,text =" Aplicar",command=lambda:[conectar(dato.get(),port),w_ip.destroy()]).place(x=200,y=40)

#Funciones para los botones

def conectar(addres,port):
    clientSocket.connect((addres, port))
    messagebox.showinfo("Mensaje Servido","Cliente conectado al robot: {0} : {1}".format(ipAddress,port))

def forward():
    clientSocket.send(bytes([ord('w')]))

def backward():
    clientSocket.send(bytes([ord('s')]))

def right():
    clientSocket.send(bytes([ord('d')]))

def left():
    clientSocket.send(bytes([ord('a')]))

def hit():
    clientSocket.send(bytes([ord('p')]))

def end_session():
    clientSocket.send(bytes([ord('q')]))

def on_release(event):
    print("Click release")
    clientSocket.send(bytes([ord(' ')]))
```

Interfaz

```
window = Tk()
window.geometry("300x300")
window.resizable(False, False)
window.title("Interfaz Alligator 3000")

fondo= resize_image("mando.png", 300, 300)
x=resize_image("x.png", 28, 37)
o=resize_image("o.png",28,37)
t=resize_image("t.png",28,37)
logo=resize_image("alligator 3000.png", 60,40)
label_fondo= Label(window, width=0, height=300, image=fondo).place(x=0,y=0)
label_logo=Label(window, image=logo,bg="gray80",bd=0).place(x=122,y=90)

#Botones en la ventana
boton_adelante = Button(None, text= "^",repeatdelay=5,repeatinterval=50,bg="gray30",bd=0,command=forward)
boton_adelante.place(x=59,y=79)
boton_atras = Button(None, text= "v",repeatdelay=5,repeatinterval=50,bg="gray30",bd=0,command=backward)
boton_atras.place(x=60,y=135)
boton_derecha = Button(None,text= ">",repeatdelay=5,repeatinterval=50,bg="gray30",bd=0,command=right)
boton_derecha.place(x=76,y=107)
boton_izquierda= Button(None, text = "<",repeatdelay=5,repeatinterval=50,bg="gray30",bd=0,command=left)
boton_izquierda.place(x=43,y=107)

boton_golpe = Button(window,command=hit, image=o,bg="gray70",bd=0).place(x=247,y=98)
boton_conectar= Button(window,text="Conectar",image=x,bg="gray70",bd=0,command=lambda:[getAddres()]).place(x=223,y=136)
boton_fin = Button(window, command= end_session, image=t, bg="gray70", bd=0).place(x=223,y=63)

boton_adelante.bind('<ButtonRelease-1>',on_release)
boton_atras.bind('<ButtonRelease-1>',on_release)
boton_derecha.bind('<ButtonRelease-1>',on_release)
boton_izquierda.bind('<ButtonRelease-1>',on_release)
ipAddress = "192.168.84.197"


# Read the command line argument for the IP address of the server
if len(sys.argv) > 2:
    print("Usage: client-laptop.py [IP-addr-of-robot]")
    sys.exit(1)
elif len(sys.argv) == 2:
    ipAddress = sys.argv[1]
    print("Using specified IP address: {}".format(ipAddress))
else:
    print("Using default IP address: {}".format(ipAddress))


clientSocket = socket.socket()
port = 8080
window.mainloop()
```

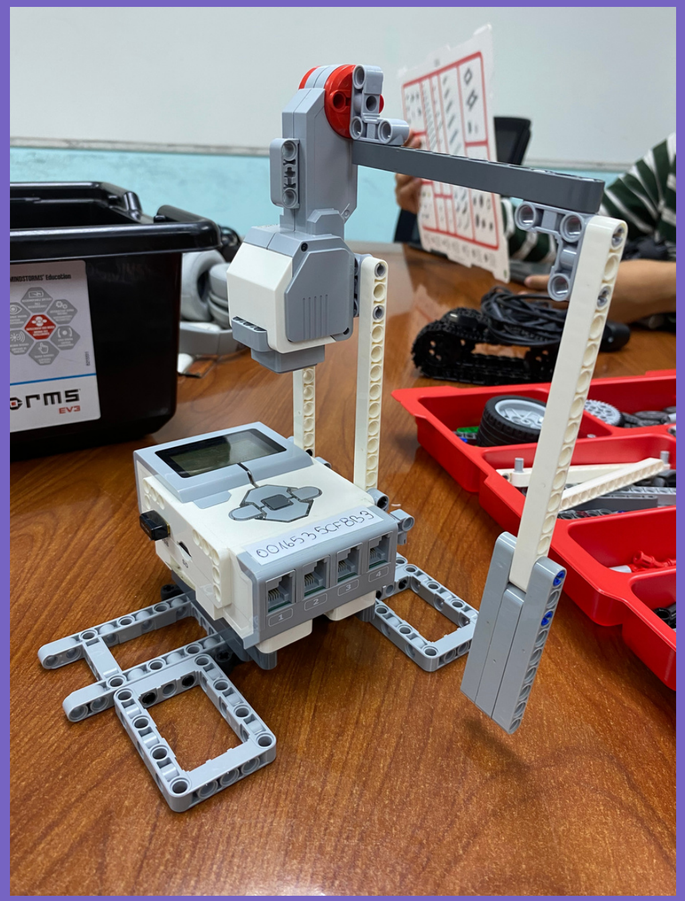
04

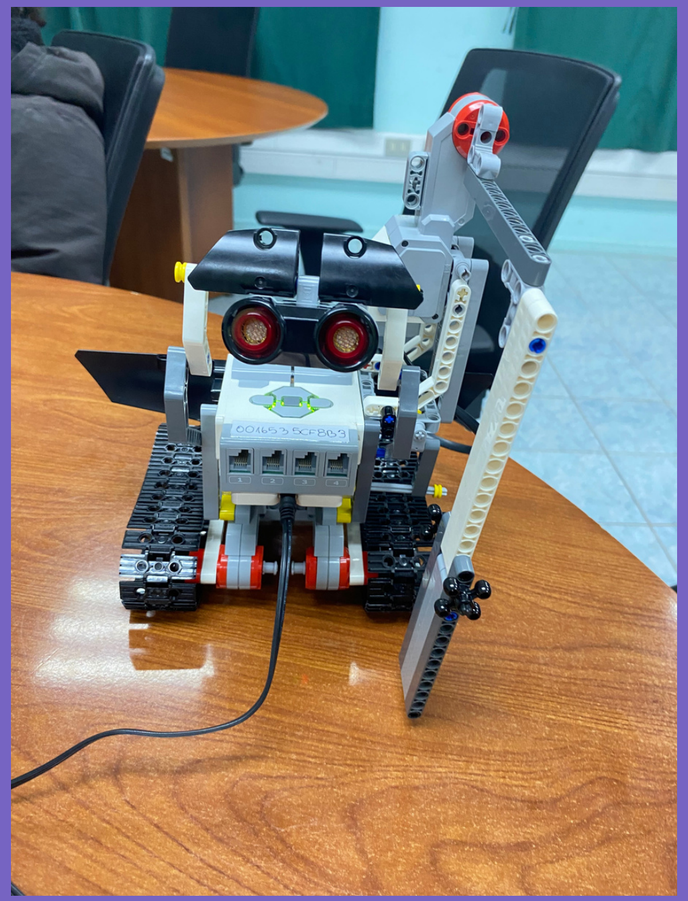
EVOLUCIÓN

Modificaciones del prototipo

PROTOTIPO I

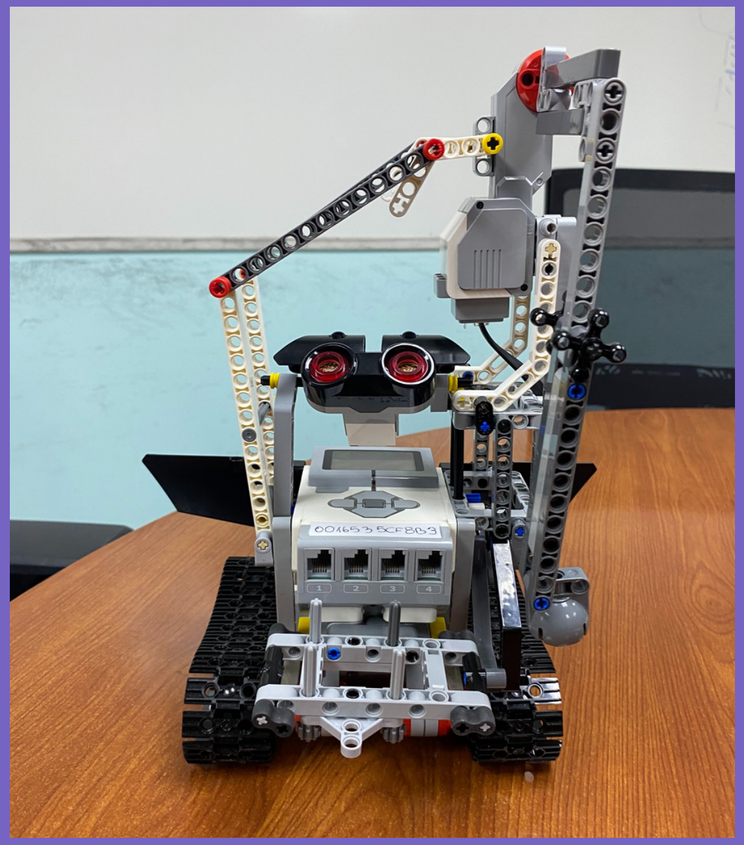
Inestabilidad en la estructura, además se dificultaba la conexión de los cables.

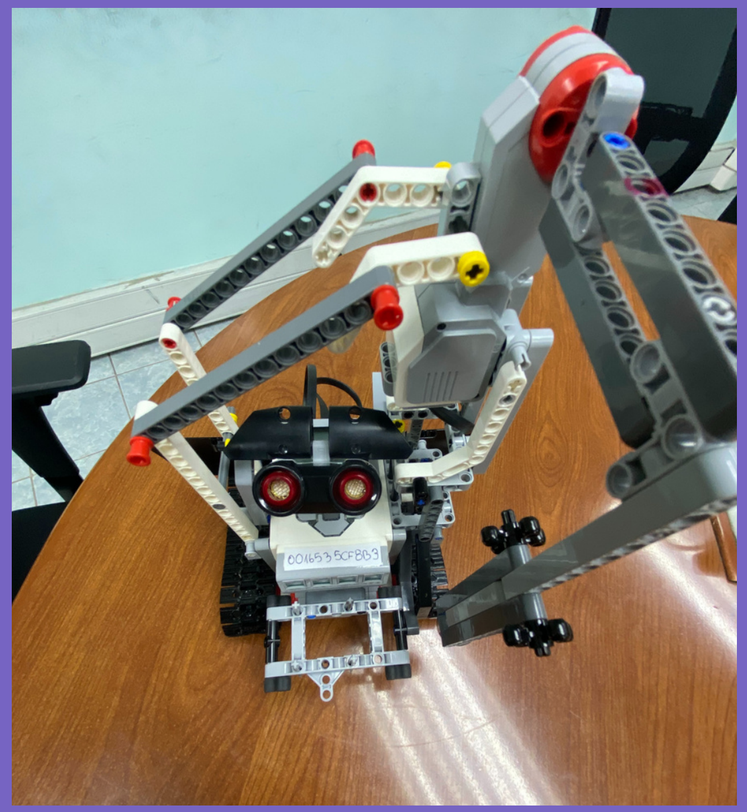


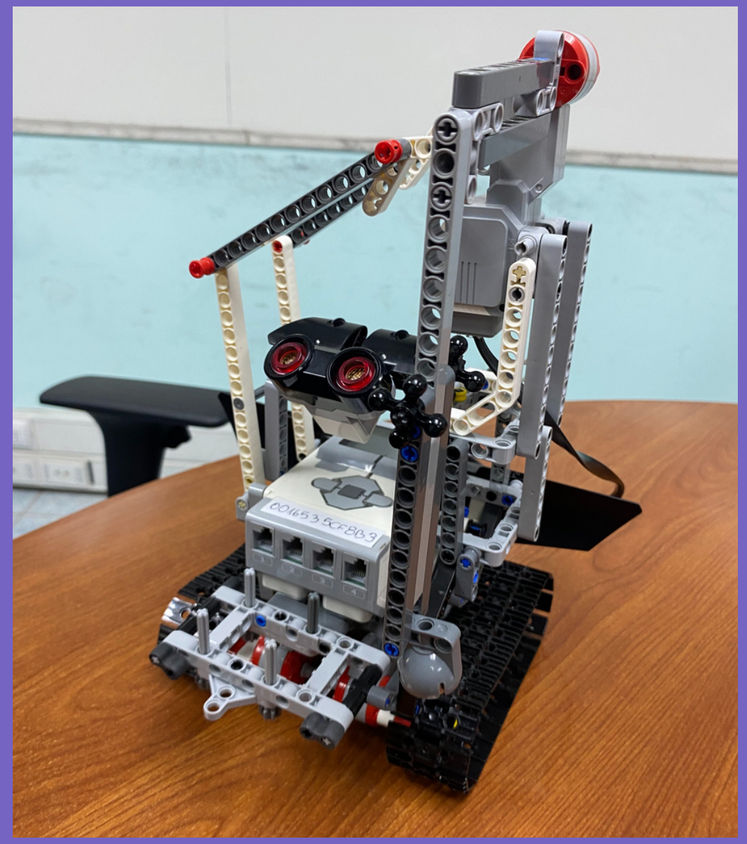


PROTOTIPO II

Modificación en el brazo, para mejor movimiento y se aplicó el refuerzo en la estructura.





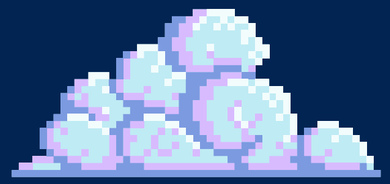


05

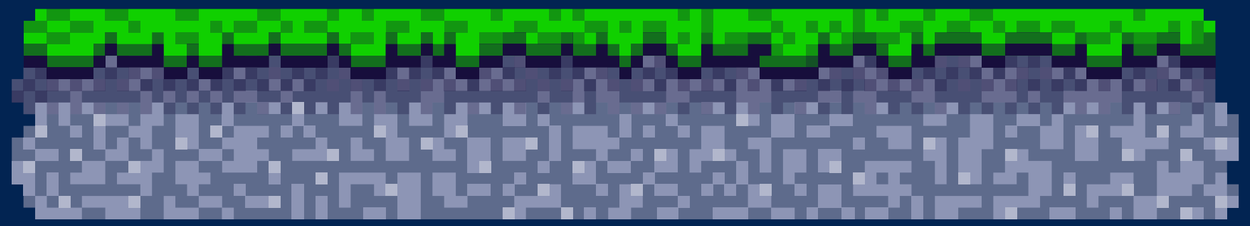
PROBLEMAS Y SOLUCIONES

Problemas encontrados	Soluciones
El robot, se encontraba descargado muchas veces.	Cargarlo al principio de la sesión de clases
La inestabilidad del robot, hizo que se desarmara cada que se ejecutaba.	Se modificaron las fuerzas del motor y se agregaron piezas para reforzar.
Incongruencias en los tiempos de la carta gantt, con la realidad del proyecto.	Se reajusto el tiempo,
El uso de una librería errónea (Pybrics) en la implementación de los movimientos del robot “Pybrics”	Se cambió de libreria, a la librería: Ev3dev2
Falta de conocimientos en el manejo de Visual Studio Code y conocimiento en redes y conexiones.	Preguntar a personas que manejan más el tema y aprender de forma autónoma en base a prueba y error.

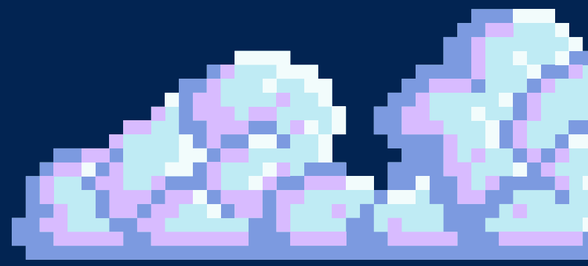
06. RESULTADOS



ESTADO ACTUAL:



- La versión finalizada del robot “Alligator 3000”
- Las funciones de movimiento.
- Interfaz gráfica.
- La creación del servidor.
- Conexión vía remota.
- La wiki del proyecto.
- Carta Gantt actualizada.
- Bitácoras, informes y presentaciones concretadas.



CONCLUSIÓN

Gracias por su atención

ALLIGATOR 3000