ROBOT “ EV3 PASCALITO ”

Integrantes:

- Denis Condori
- Esteban Gutierrez
- Ignacio Gallardo
- Fernando Klinger
- Martin Salinas

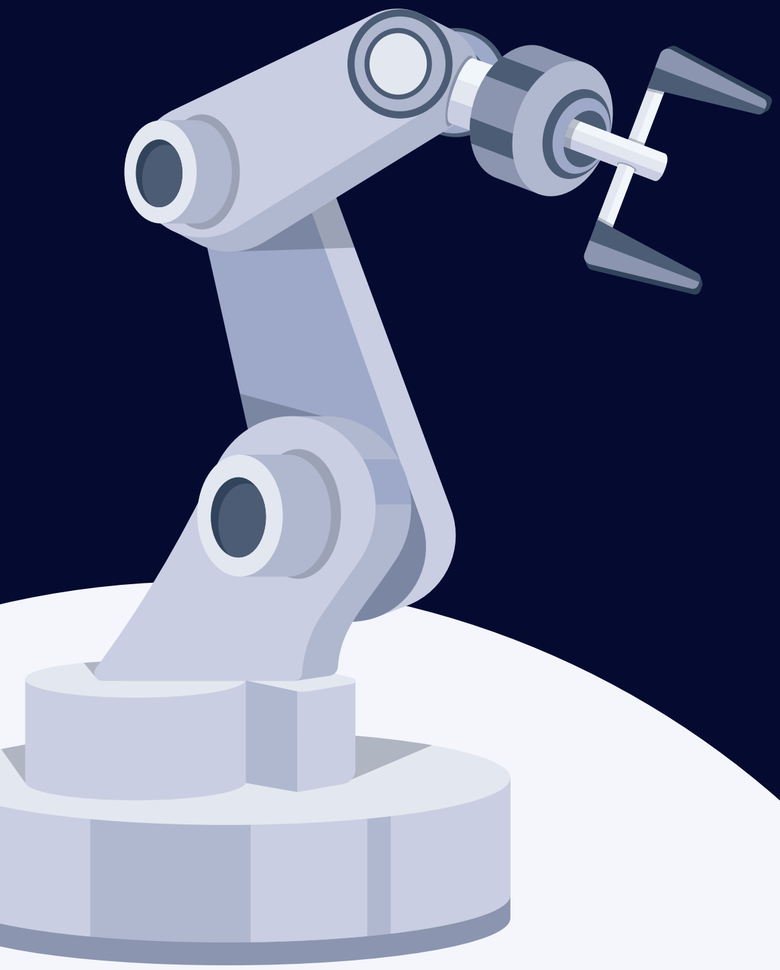
Profesor: Humberto Urrutia

Asignatura: Proyecto I

INTRODUCCIÓN

En esta fase intermedia de nuestro proyecto, nos complace presentarles los notables avances y mejoras realizadas en nuestro robot "Pascalito". A lo largo de este período, hemos trabajado en la optimización de sus requerimientos, la refinación de su arquitectura, la interfaz y el código del robot. Además, compartiremos las evoluciones significativas experimentadas por el robot y proporcionaremos una visión detallada del estado actual del proyecto.

OBJETIVOS



Objetivos General

Diseñar y construir un robot utilizando el kit de Lego Mindstorms EV3 con la capacidad de desplazarse y golpear una pelota, imitando las acciones de un jugador de golf. Este robot será controlado por un usuario.

Objetivos Específicos

- Crear un servidor que permita la conexión del cliente con el robot.
- Aplicar la teoría física del tiro parabólico.
- Crear una interfaz eficiente y amigable con el usuario.

ETAPAS DEL PROYECTO

Primera Etapa

Segunda Etapa

Estado Actual

REASIGNACIÓN DE ROLES

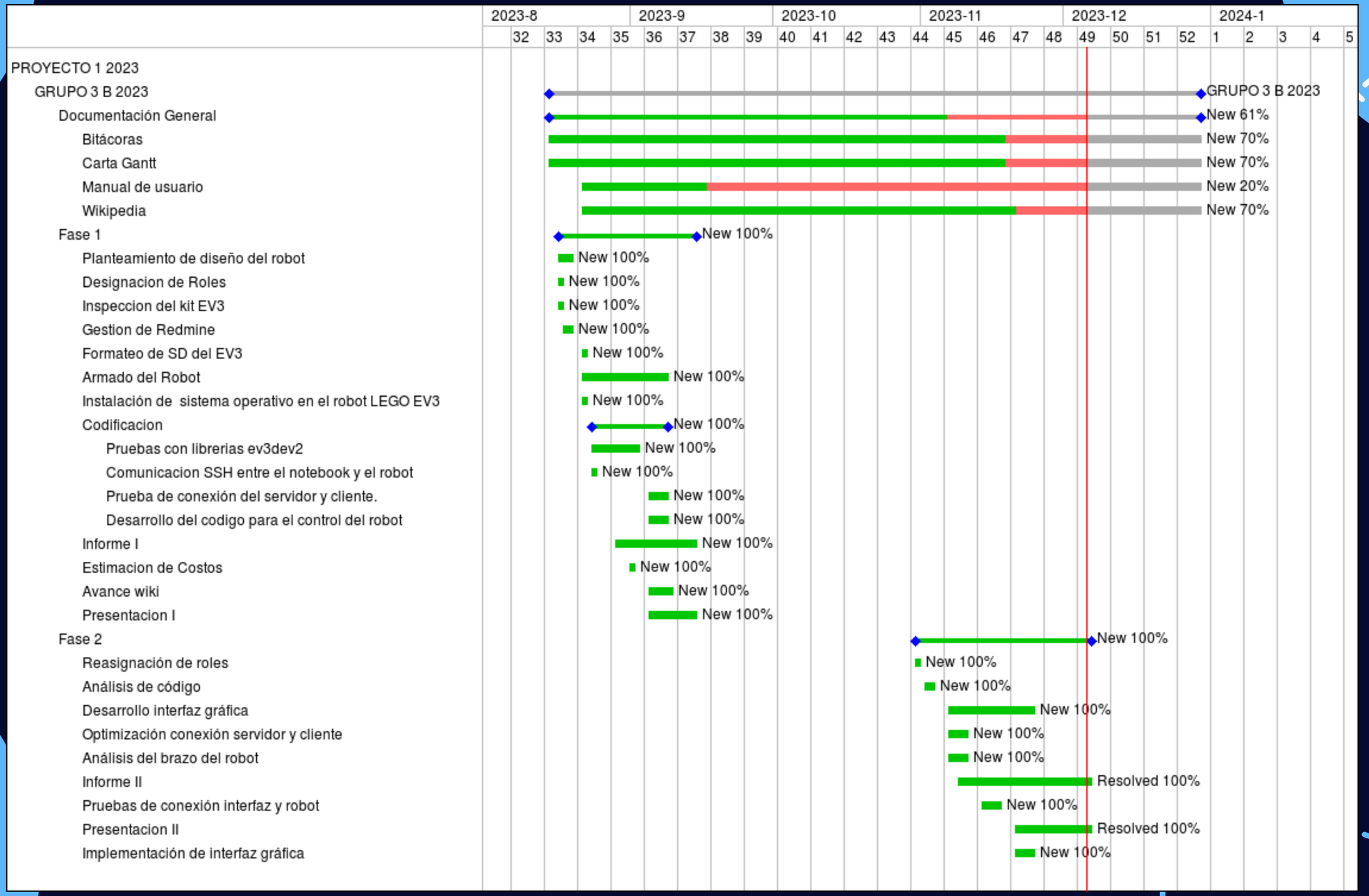
ASIGNACIÓN DE ROLES ANTERIORES

Programador(es)	-Ignacio Gallardo -Esteban Gutierrez
Ensamblador	-Martin Salinas
Documentador	-Fernando Klinger
Diseñador	-Denis Condori
Jefe de Grupo	-Ignacio Gallardo

ASIGNACIÓN DE ROLES ACTUAL

Programador	- Denis Condori
Ensamblador	-Fernando Klinger
Documentador	-Ignacio Gallardo
Diseñador	-Esteban Gutierrez
Jefe de Grupo	-Martin Salinas

CARTA GANTT


2023-8
2023-9
2023-10
2023-11
2023-12
2024-1
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 1 2 3 4 5
PROYECTO 1 2023
GRUPO 3 B 2023
Documentación General
Bitácoras
Carta Gantt
Manual de usuario
Wikipedia
Fase 1
Planteamiento de diseño del robot
Designacion de Roles
Inspeccion del kit EV3
Gestion de Redmine
Formateo de SD del EV3
Armado del Robot
Instalación de sistema operativo en el robot LEGO EV3
Codificacion
Pruebas con librerias ev3dev2
Comunicacion SSH entre el notebook y el robot
Prueba de conexión del servidor y cliente.
Desarrollo del codigo para el control del robot
Informe I
Estimacion de Costos
Avance wiki
Presentacion I
Fase 2
Reasignación de roles
Análisis de código
Desarrollo interfaz gráfica
Optimización conexión servidor y cliente
Análisis del brazo del robot
Informe II
Pruebas de conexión interfaz y robot
Presentacion II
Implementación de interfaz gráfica
GRUPO 3 B 2023
New 61%
New 70%
New 70%
New 20%
New 70%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
New 100%
Resolved 100%
New 100%
Resolved 100%
New 100%

REQUERIMIENTOS FUNCIONALES

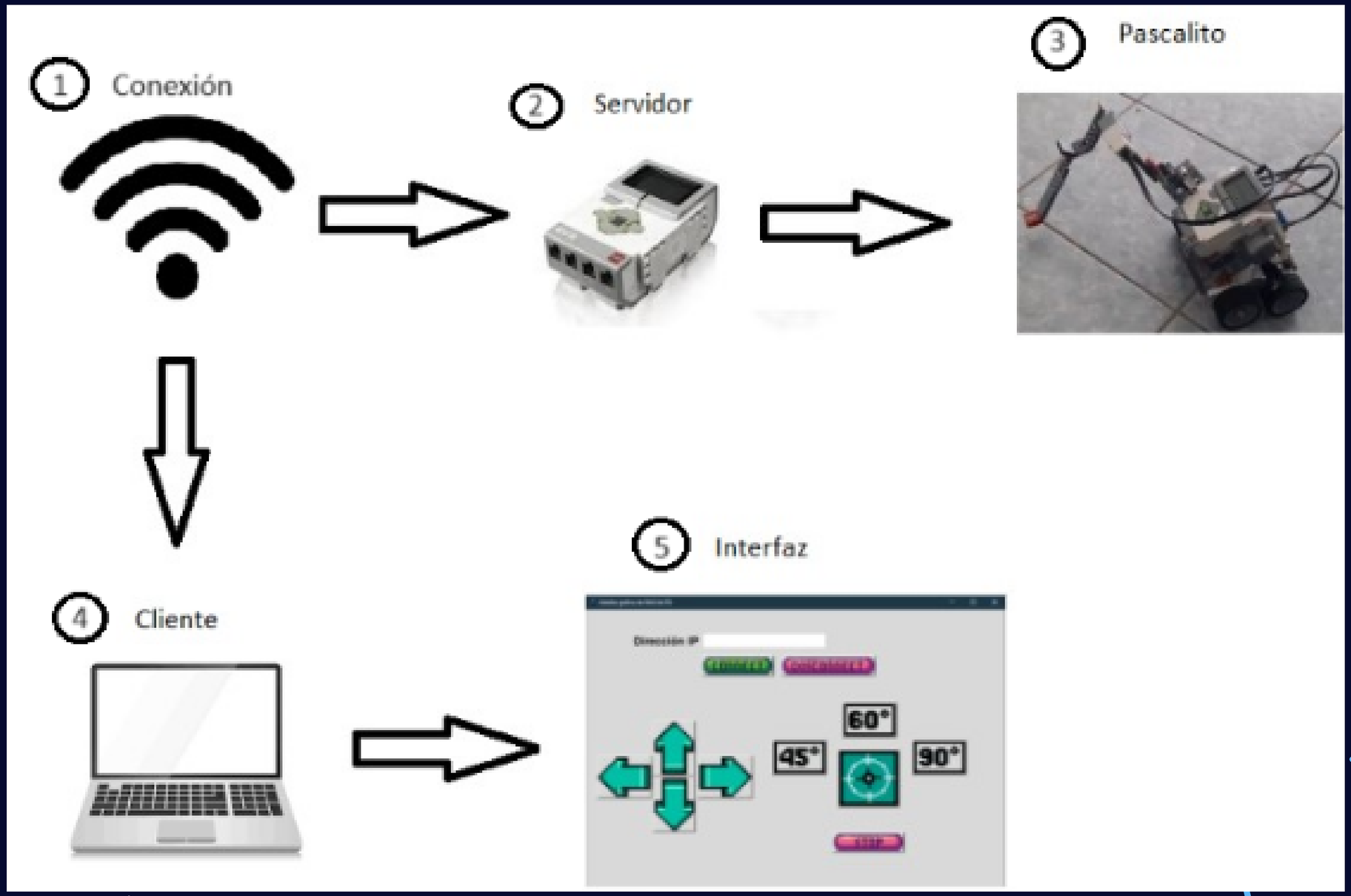
- El robot tiene que tener la capacidad de golpear una pelota con una estructura que imita a un palo.
- Debe tener la capacidad de moverse en todas las direcciones de manera fluida.
- La operación del robot estará bajo el control directo del usuario a través de un programa dedicado.
- Se requiere un servidor capaz de comunicar al programa con el usuario.

REQUERIMIENTOS NO FUNCIONALES

- La programación del software debe llevarse a cabo en el lenguaje Python y en el entorno del sistema operativo Linux

- La interfaz debe ser eficiente y amigable con el usuario.

ARQUITECTURA



SERVIDOR

- **Librerías**

```
1    import socket
2    from ev3dev2.motor import MediumMotor, OUTPUT_A, OUTPUT_B, OUTPUT_C, SpeedPercent, MoveTank
```

- **Establecemos los puertos de los actuadores**

```
4    tankmoves = MoveTank(OUTPUT_B,OUTPUT_C)
5    brazo = MediumMotor(OUTPUT_A)
```

SERVIDOR

- **Iniciando el servidor usando la librería socket**

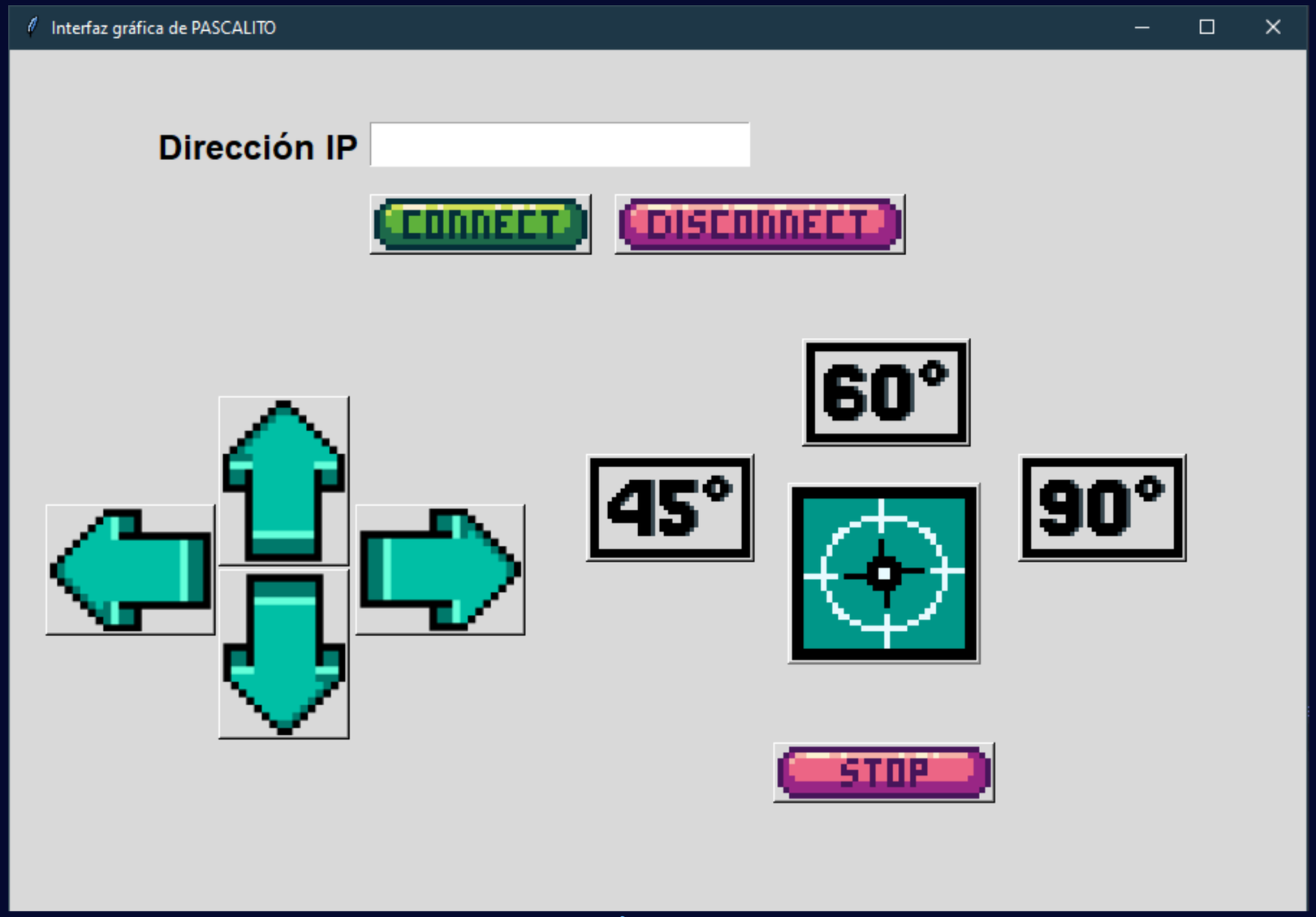
```
7   ip_host = input("Ingrese la ip que muestra el robot: ")
8   host = ip_host
9   port = 8000
10  server = socket.socket()
11  server.bind((host, port))
12  server.listen(5)
13  conexion, addr = server.accept()
14  print("Nueva conexion establecida")
15  print("conectado a ", addr)
```

SERVIDOR

Bucle de funcionamiento de Pascalito

```
17  running = True
18
19  while running:
20      mensaje = conexion.recv(1024).decode("utf-8")
21      print(mensaje)
22      if mensaje == 'arriba':
23          tankmoves.on(SpeedPercent(10), SpeedPercent(10))
24      elif mensaje == 'abajo':
25          tankmoves.on(SpeedPercent(-10), SpeedPercent(-10))
26      elif mensaje == 'izquierda':
27          tankmoves.on(SpeedPercent(20), SpeedPercent(-20))
28      elif mensaje == 'derecha':
29          tankmoves.on(SpeedPercent(-20), SpeedPercent(20))
30      elif mensaje == 'detener':
31          tankmoves.on(SpeedPercent(0), SpeedPercent(0))
32      elif mensaje == 'apagar':
33          running = False
34          conexion.close()
35          print("La conexion termino!")
36      elif mensaje == 'golpear':
37          brazo.on_for_degrees(SpeedPercent(30), 45)
38          brazo.on_for_degrees(SpeedPercent(100), -95)
```

INTERFAZ


Interfaz gráfica de PASCALITO
Dirección IP
CONNECT
DISCONNECT
60°
45°
90°
STOP

INTERFAZ

- **Librerías**

```
1    import tkinter as tk
2    import socket
```

- **Creamos la ventana**

```
36   # Crea la ventana principal
37   ventana = tk.Tk()
38   ventana.title("Interfaz gráfica de PASCALITO")
39   ventana.geometry("900x600")
40   ventana.config(bg="grey85")
```

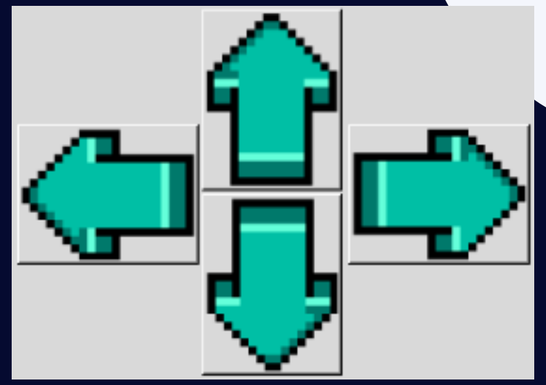
```
99   # Iniciar el bucle principal
100  ventana.mainloop()
```

INTERFAZ

- Conexiones


```
42  # Etiqueta y entrada para la dirección IP
43  etiqueta_ip = tk.Label(ventana, text="Dirección IP", font=("Arial",18,"bold"),bg="grey85")
44  etiqueta_ip.place(x=100, y=50)
45
46  entrada_ip = tk.Entry(ventana, font=("bold",18))
47  entrada_ip.place(x=250, y=50)
48
49  # Creamos los botones para generar la conexion y para terminar la conexion
50  img_boton_conectar= tk.PhotoImage(file="connect.png")
51  boton_conectar = tk.Button(ventana, text="Conectar",image=img_boton_conectar,bg="grey85", command=conectar)
52  boton_conectar.place(x=250, y=100)
53
54  img_boton_desconectar= tk.PhotoImage(file="disconnect.png")
55  boton_desconectar = tk.Button(ventana, text="Desconectar",image=img_boton_desconectar,bg="grey85", command=apagar)
56  boton_desconectar.place(x=420, y=100)
```

INTERFAZ



- **Controles de movimiento**

```
81    # Botones de movimiento
82    img_boton_arriba = tk.PhotoImage(file="flecha-up1.png")
83    boton_arriba = tk.Button(ventana, image=img_boton_arriba,bg="grey85",command=lambda: mover("arriba"))
84    boton_arriba.place(x=145, y=240)
85
86    img_boton_izquierda = tk.PhotoImage(file="flecha-izquierda.png")
87    boton_izquierda = tk.Button(ventana, text="Izquierda",image=img_boton_izquierda,bg="grey85" ,command=lambda: mover("izquierda"))
88    boton_izquierda.place(x=25, y=315)
89
90
91    img_boton_derecha = tk.PhotoImage(file="flecha-derecha.png")
92    boton_derecha = tk.Button(ventana, text="Derecha",image=img_boton_derecha,bg="grey85", command=lambda: mover("derecha"))
93    boton_derecha.place(x=240, y=315)
94
95    img_boton_abajo = tk.PhotoImage(file="flecha-abajo.png")
96    boton_abajo = tk.Button(ventana, text="Abajo",image=img_boton_abajo,bg="grey85", command=lambda: mover("abajo"))
97    boton_abajo.place(x=145, y=360)
```

INTERFAZ

- **Controles de disparo y ángulo**


```
59  # Botones de disparo
60  img_boton_45grados= tk.PhotoImage(file="45grados.png")
61  boton_45grados= tk.Button(ventana, text="45°",image=img_boton_45grados,bg="grey85",command=angulo45)
62  boton_45grados.place(x=400, y=280)
63
64  img_boton_60grados= tk.PhotoImage(file="60grados.png")
65  boton_60grados= tk.Button(ventana, text="60°",image=img_boton_60grados,bg="grey85",command=angulo60)
66  boton_60grados.place(x=550, y=200)
67
68  img_boton_90grados= tk.PhotoImage(file="90grados.png")
69  boton_90grados= tk.Button(ventana, text="90°",image=img_boton_90grados,bg="grey85",command=angulo90)
70  boton_90grados.place(x=700, y=280)
71
72  img_boton_disparar= tk.PhotoImage(file="shot.png")
73  boton_disparar= tk.Button(ventana, text="Disparar",image=img_boton_disparar,command=disparar)
74  boton_disparar.place(x=540, y=300)
```

INTERFAZ

- **Boton detener**


```
76  # Boton para detener el movimiento
77  img_boton_stop= tk.PhotoImage(file="STOP.png")
78  boton_detener = tk.Button(ventana, text="Detener",image=img_boton_stop,bg="grey85",command=detener)
79  boton_detener.place(x=530, y=480)
```

FUNCIONES CLIENTE

- **Función movimiento**

```
4    def mover(direccion):
5        global mi_socket
6        if direccion == "arriba":
7            mi_socket.send("arriba".encode("utf-8"))
8        elif direccion == "abajo":
9            mi_socket.send("abajo".encode("utf-8"))
10       elif direccion == "izquierda":
11           mi_socket.send("izquierda".encode("utf-8"))
12       elif direccion == "derecha":
13           mi_socket.send("derecha".encode("utf-8"))
```

FUNCIONES CLIENTE

- **Función de conexión**

```
15  def conectar():
16      global mi_socket
17      # Función para manejar la conexión con la dirección IP ingresada
18      ip = entrada_ip.get()
19      host = str(ip)
20      port = 8000
21      mi_socket = socket.socket()
22      mi_socket.connect((host, port))
23
24  def apagar():
25      global mi_socket
26      mi_socket.send("apagar".encode("utf-8"))
```

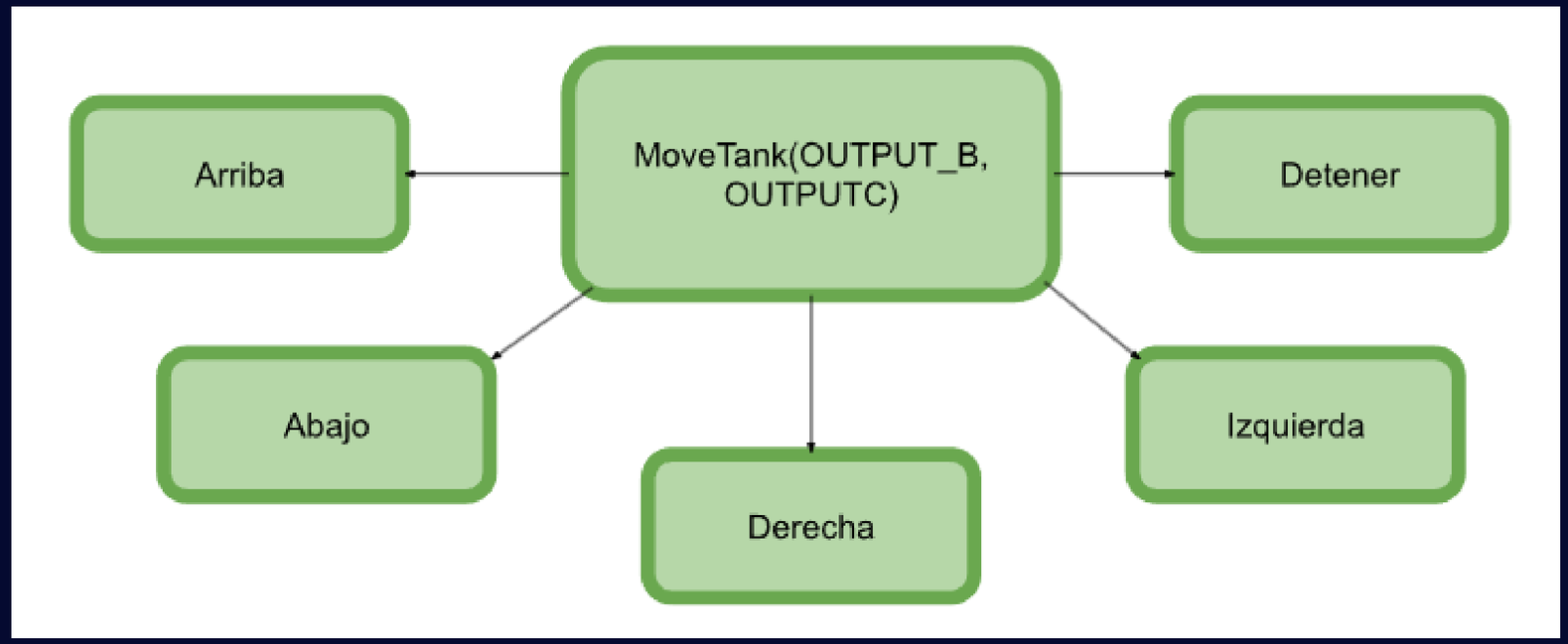
FUNCIONES CLIENTE

- **Función movimiento**

```
32  def angulo45():
33      global angulo
34      angulo = 45
35
36  def angulo60():
37      global angulo
38      angulo = 60
39
40  def angulo90():
41      global angulo
42      angulo = 90
43
44  def disparar():
45      global mi_socket
46      global angulo
47      if angulo == 45:
48          mi_socket.send("disparar45".encode("utf-8"))
49      elif angulo == 60:
50          mi_socket.send("disparar60".encode("utf-8"))
51      elif angulo == 90:
52          mi_socket.send("disparar90".encode("utf-8"))
```

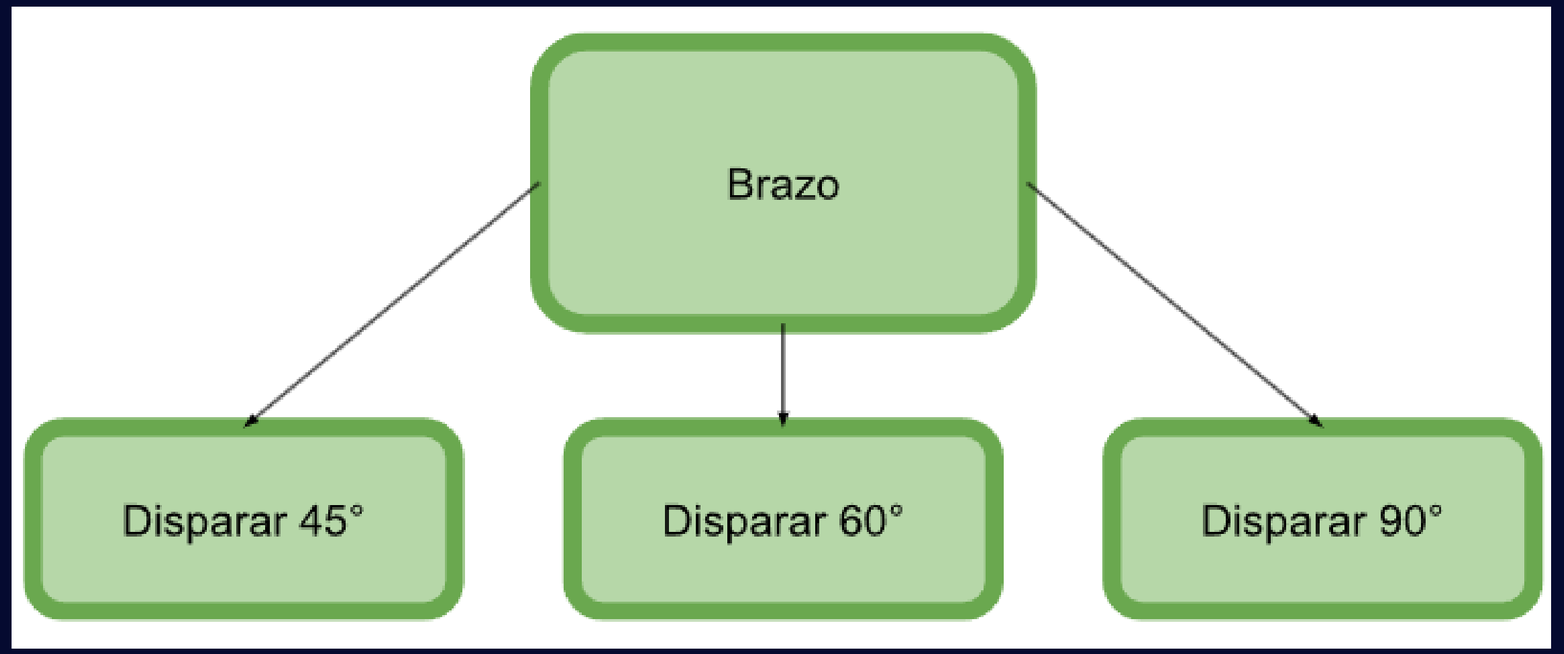
DIAGRAMAS

- **Movimiento**



DIAGRAMAS

- **Brazo**

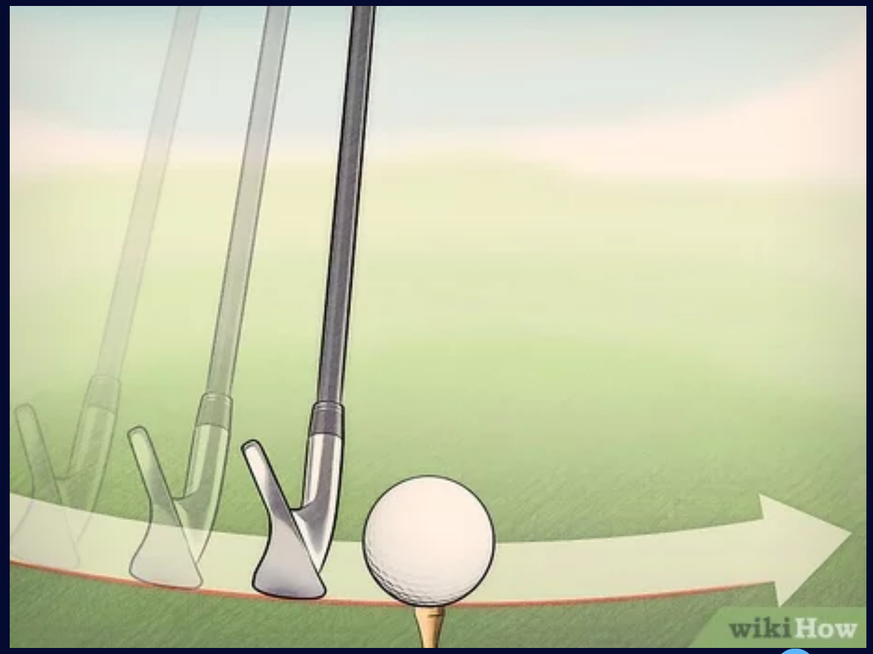


FÍSICA DETRAS DEL GOLPE DE GOLF

- Leyes de Newton
- Movimiento parabólico
- Fuerza de roce

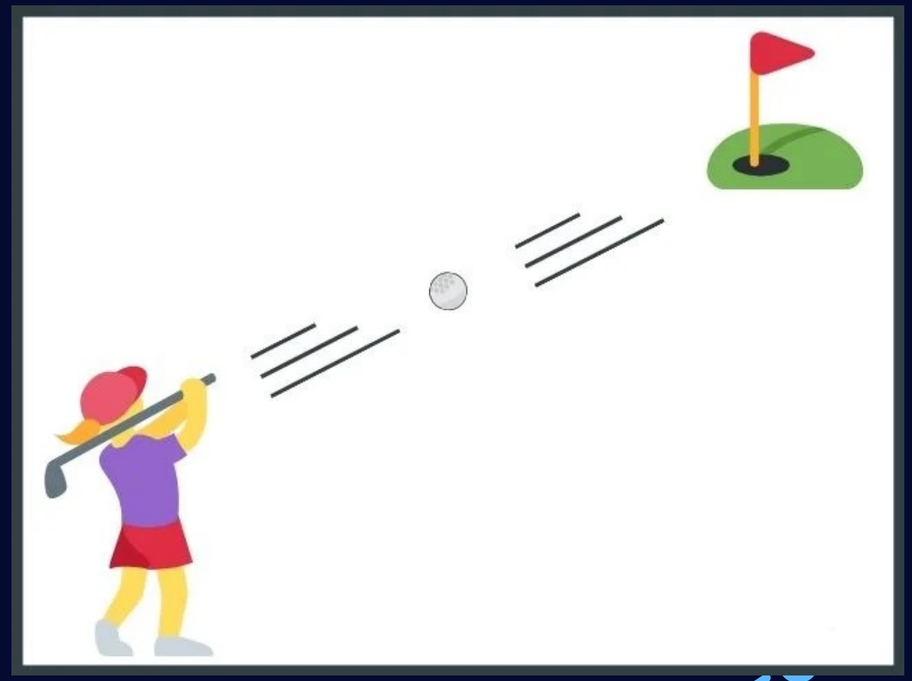
LEYES DE NEWTON

- **Primera ley de Newton (Ley de inercia)**



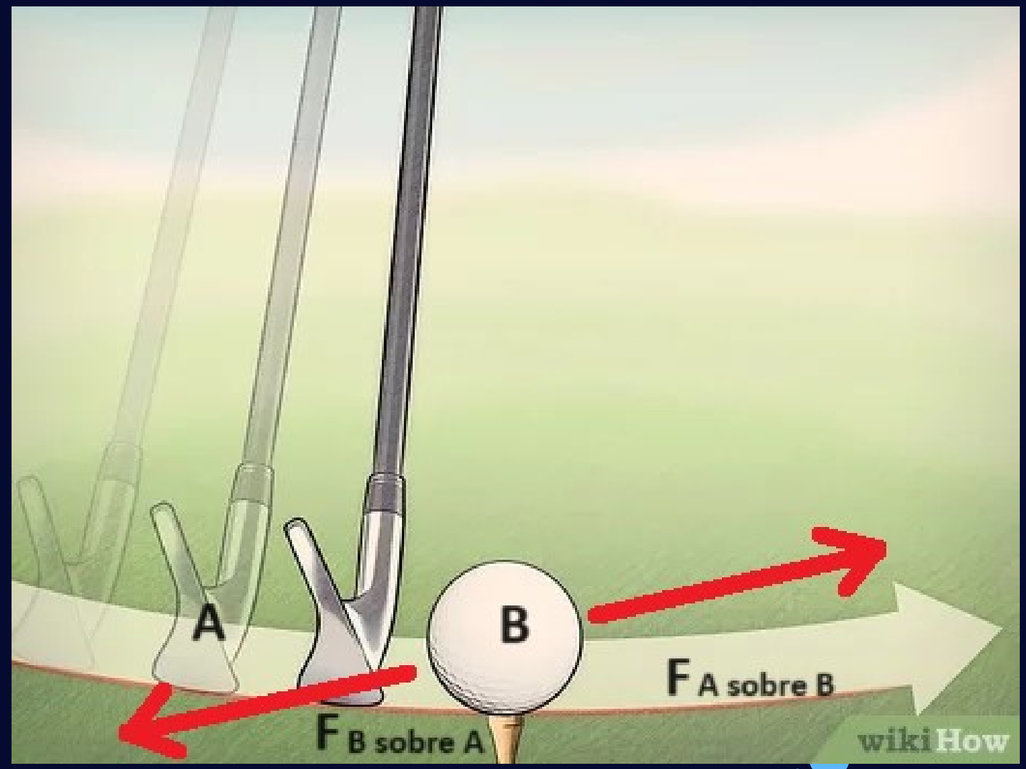
LEYES DE NEWTON

- **Segunda ley de Newton (Fuerza y aceleración)**



LEYES DE NEWTON

- **Tercera ley de Newton (Acción y reacción)**

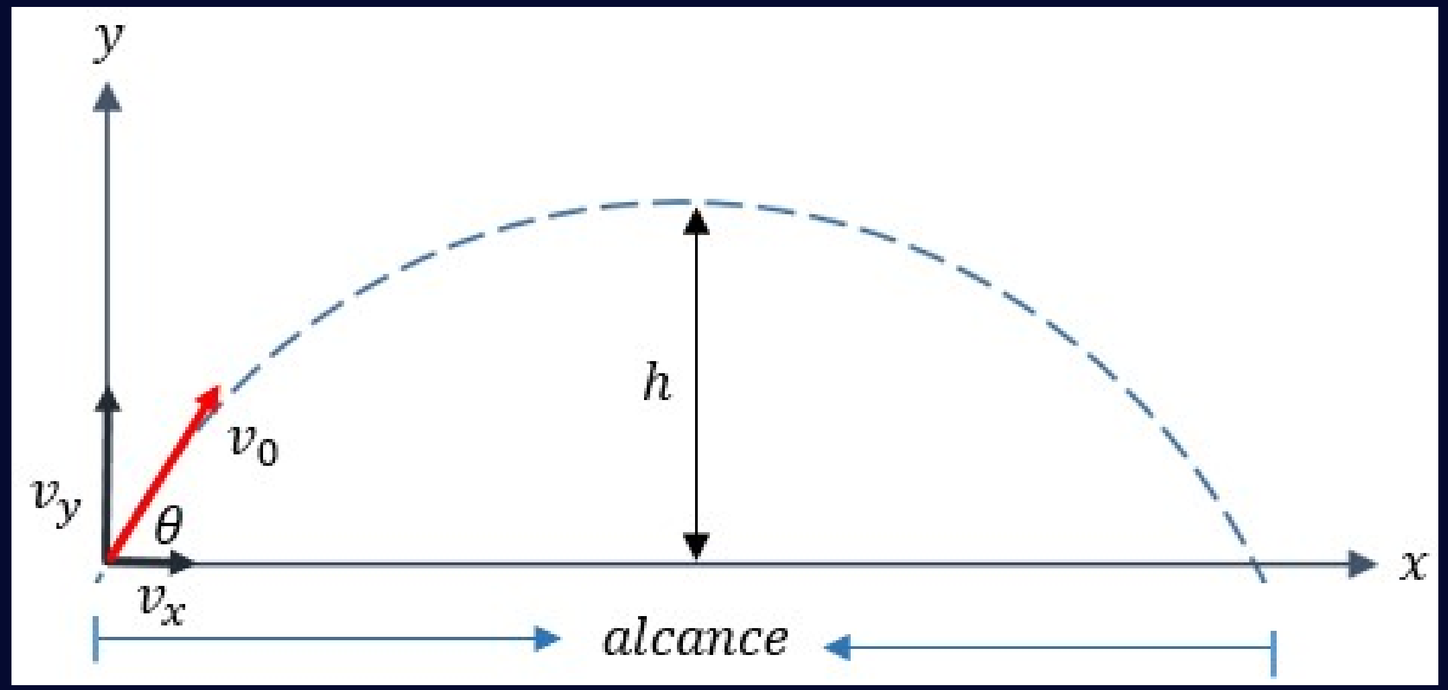


TEORÍA SOBRE EL TIRO DE PROYECTILES

$$Vo_y = Vo\ sin(\theta)$$

$$Vo_x = Vo\ cos(\theta)$$

TEORÍA SOBRE EL TIRO DE PROYECTILES



$$Vo_x = Vo\ cos(\theta)$$

$$Vo_y = Vo\ sin(\theta)$$

Movimiento en X

$$X = Vo\ cos(\theta) \cdot t$$

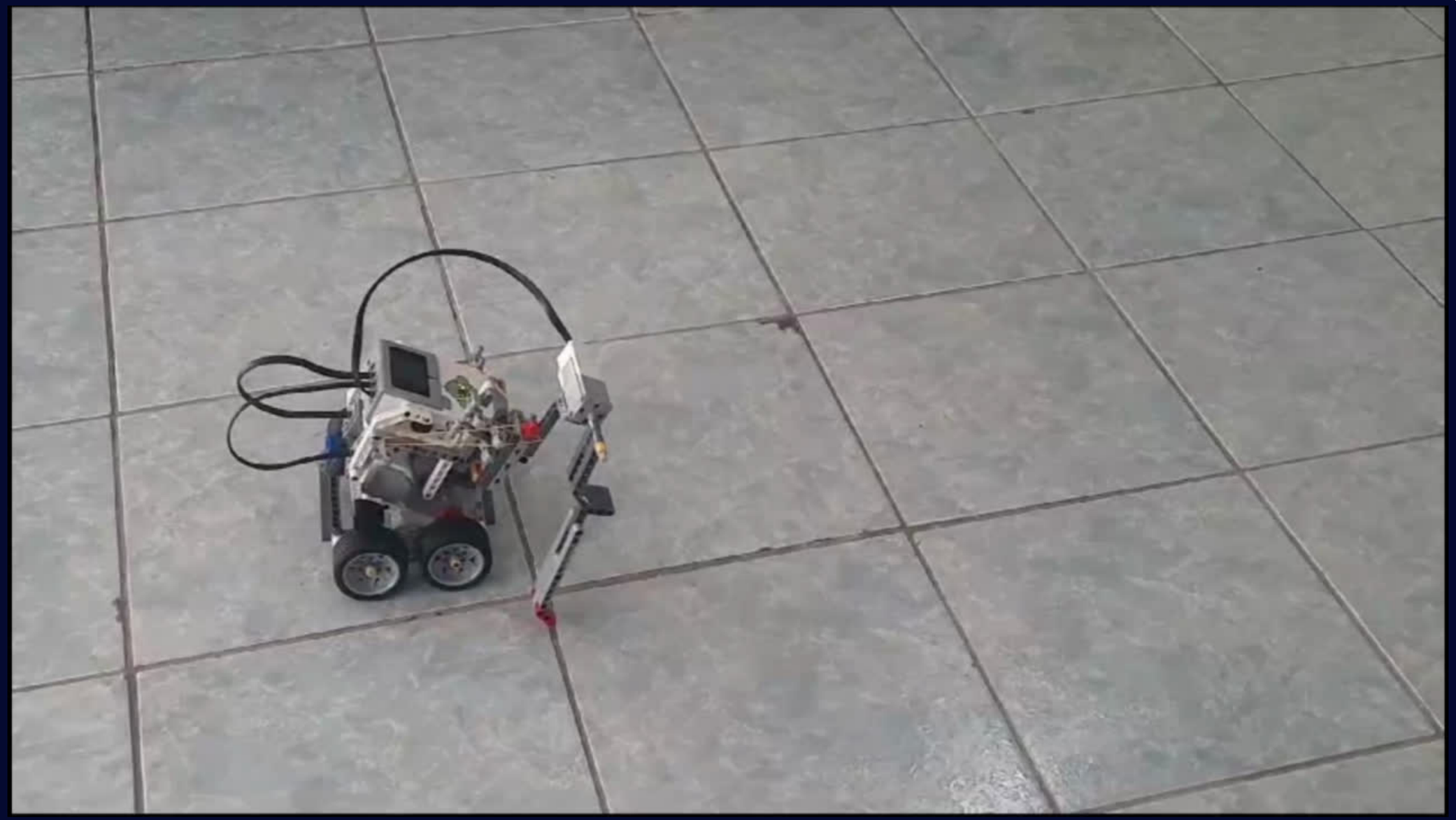
Movimiento en Y

$$Y = Vo\ sin(\theta) \cdot t - \frac{1}{2} g t^2$$

FUERZA DE ROCE

Estado actual del proyecto

1. **Estructura**
2. **Movimiento**
3. **Interfaz gráfica**
4. **Servidor**
5. **Conexión remota**



Problemas y soluciones

Problemas encontrados	Soluciones
Problema estructural debido a que se ladea el brazo.	Reforzar o cambiar la estructura del brazo de PASCALITO
Problemas con el cambio de ángulos de disparo.	Fabricar un tee de golf para distintos ángulos.
Problema de conexión del robot al reiniciar la conexión.	Hacer un debug en el código del servidor hasta poder arreglar el bug.
Falta de poder en el disparo.	Agregar algún mecanismo con engranajes para aumentar la potencia del disparo.

CONCLUSIÓN

En esta segunda fase de nuestro robot "Pascalito", enfrentamos desafíos que nos llevaron a reconstruir el robot en dos ocasiones debido a inconvenientes relacionados con su funcionalidad, por ejemplo errores en las piezas y problemas de equilibrio. A pesar de estos contratiempos, nuestro equipo demostró su capacidad para superar obstáculos y aprender de nuestras experiencias. Logramos superar los problemas encontrados y estamos cerca de finalizar con éxito nuestro proyecto. Este proceso nos ha permitido fortalecer nuestras habilidades individuales y colectivas, destacando nuestra determinación para alcanzar nuestros objetivos a pesar de los contratiempos iniciales.

MUCHAS GRACIAS

por ver esta presentación