UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

TIGER BOT
COMPANY

AVANCE TIGER BOT

Proyecto I

Integrantes:

-Diego López
-Gustavo Morales
-Sebastián Becerra
-Bryan Vega
-Sergio Huanca

Profesor:

-Humberto Urrutia

Introducción

Asignación de roles

Jefe de Grupo	Programador	Ensamblador	Diseñador	Documentador
Encargado de representar al equipo de trabajo, de la organización y de la toma de decisiones.	Encargado de desarrollar e implementar el código en Python logrando así que el robot pueda ejecutar las acciones solicitadas.	Encargado de diseñar y armar el robot de tal manera que pueda moverse en todas las direcciones y lance un proyectil.	Encargado de la estética de la interfaz gráfica.	Encargado de realizar los informes, presentaciones, bitácoras, video, manual de usuario y wiki del proyecto
Diego López	Diego López y Sebastián Becerra	Gustavo Morales	Diego López	Sergio Huanca, Sebastián Becerra y Bryan Vega

Objetivos

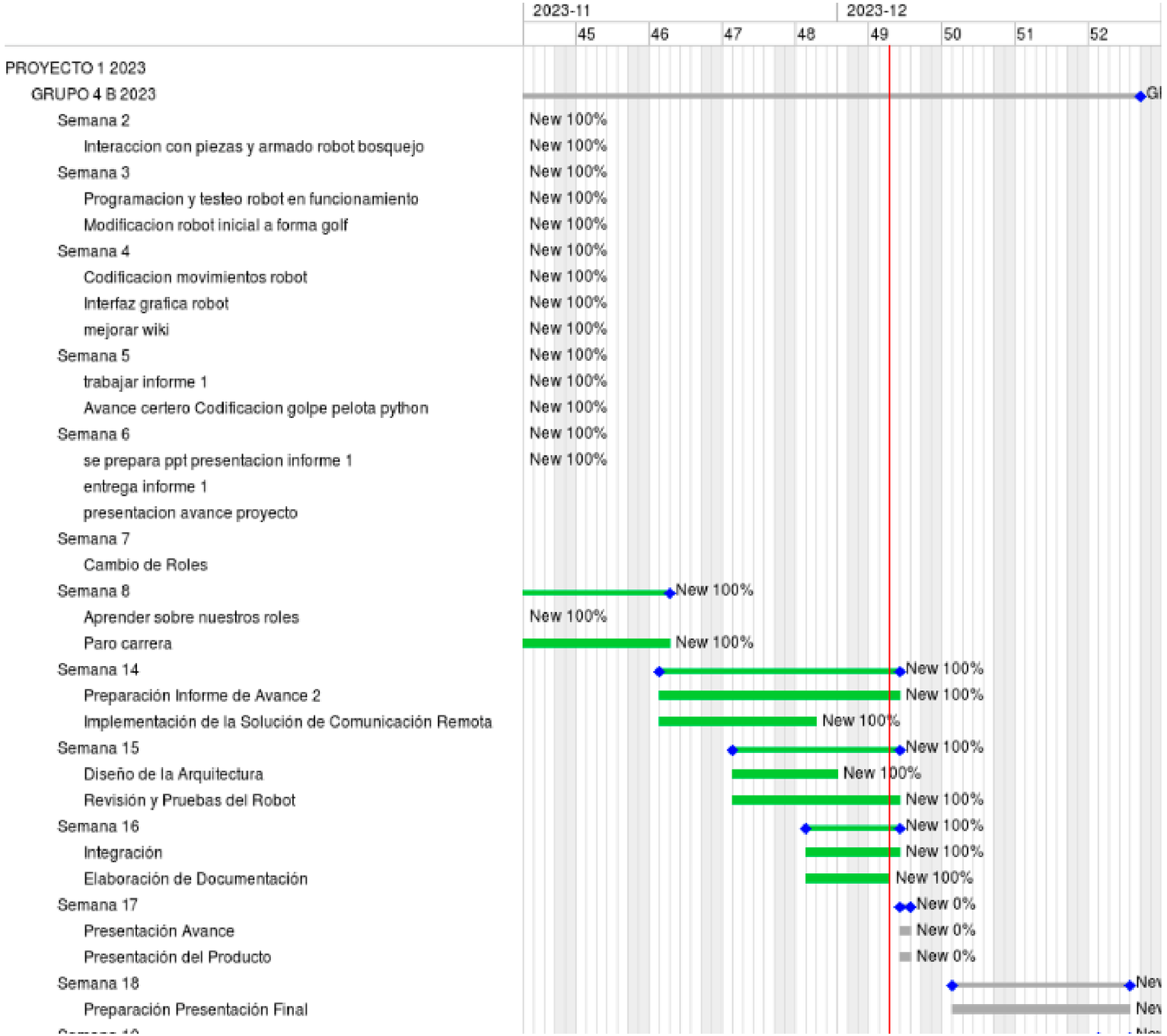
OBJETIVO GENERAL

- Diseñar y armar un robot con el kit Lego Mindstorms EV3 que pueda jugar golf, controlado a través de una interfaz gráfica remota usando Python.

OBJETIVOS ESPECIFICOS

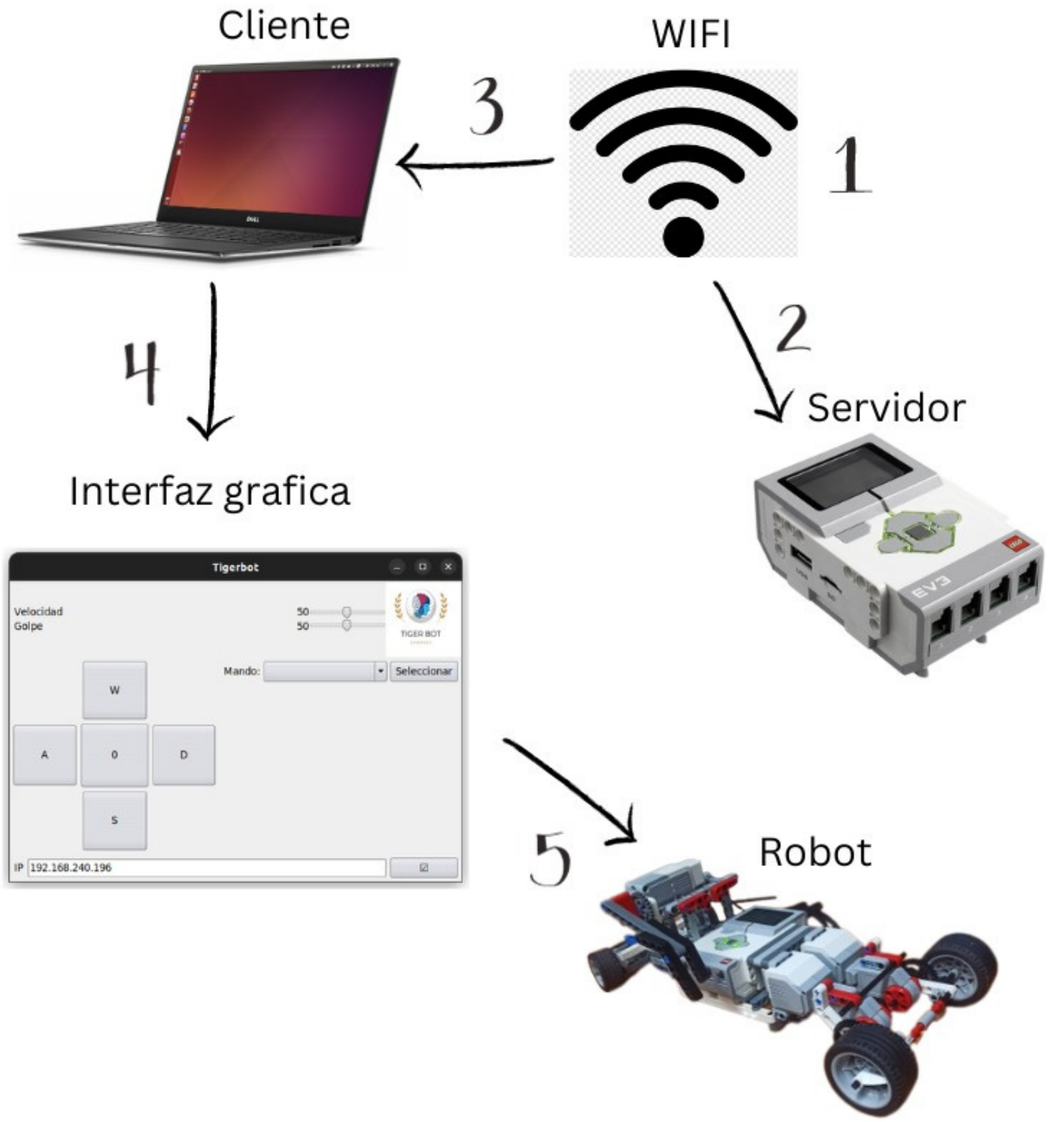
- Armar un prototipo funcional del robot que golpee una pelota y llegue a un objetivo aproximado.
- Estudiar módulo tkinter para escribir interfaz gráfica.
- Estudiar módulo socket para establecer comunicación remota.
- Aplicar conocimientos en mecánica clásica para predecir el tiro de la pelota.
- Gestionar tiempos de trabajo para maximizar tiempo.
- Concretar el informe avance.
- Efectuar presentación del informe avance.

Asignación de Tiempo

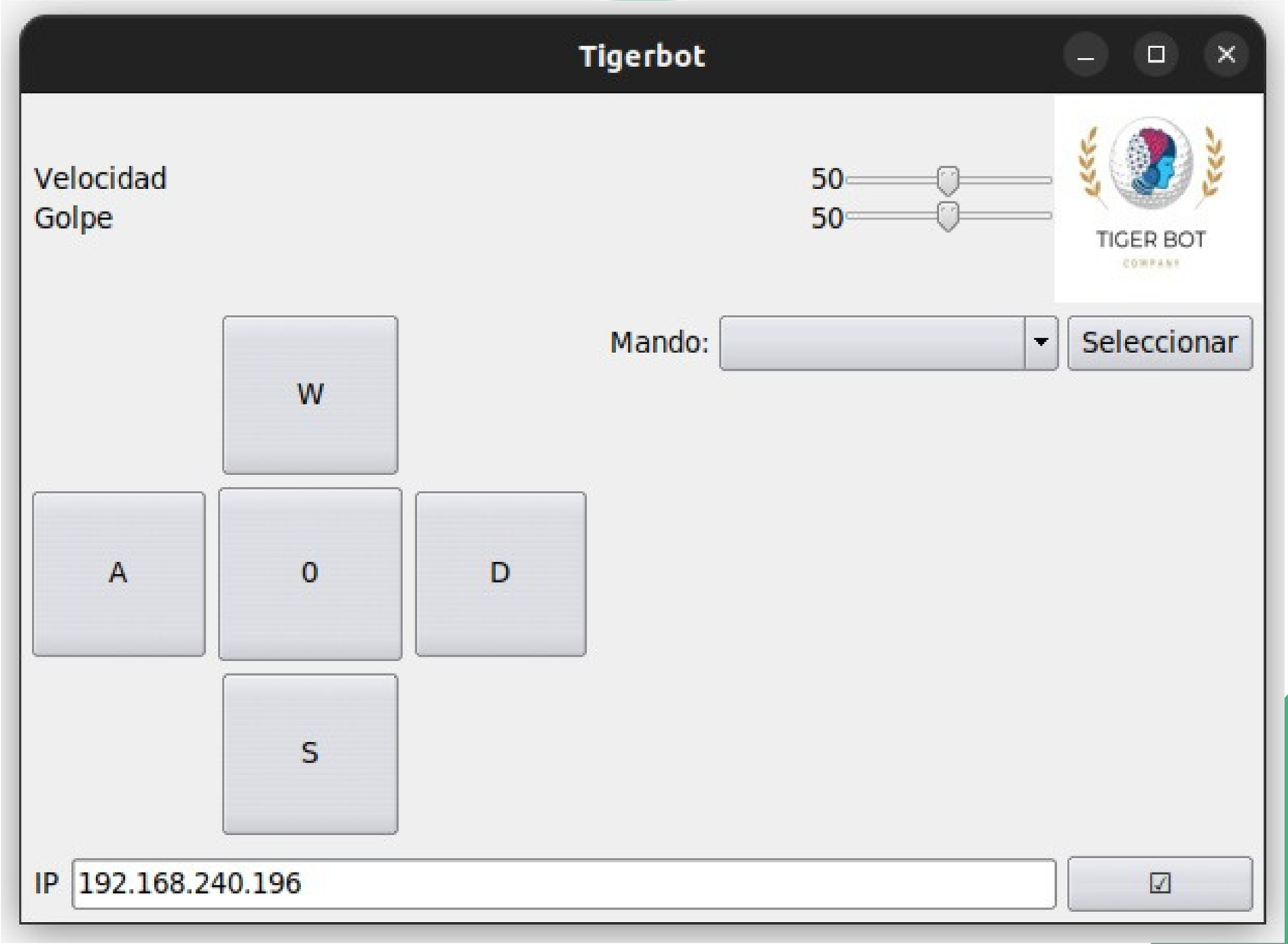


Análisis y Diseño

Arquitectura



Interfaz


Tigerbot
Velocidad
50
Golpe
50
TIGER BOT
Mando:
Seleccionar
W
A
0
D
S
IP
192.168.240.196

Cliente

```python
@hilo
def crear_robot():
    try:
        self.robot = Robot(self.ip['entry'].get())
        print("Conexión exitosa")
    except:
        self.ip['entry'].delete(0, tk.END)
        self.ip['entry'].insert(0, "Error de Conexión")
        print("Error de Conexión")
```

```python
import socket

PORT    = 2334
SIZE    = 32

class Robot:
    def __init__(self, HOST):
        self.s = socket.create_connection((HOST, PORT), 5)
        self.s.settimeout(None)

    def avanzar(self, velocidad):
        self.s.send(bytes(f'{f"avanzar {velocidad}":{SIZE}}', "utf-8"))

    def retroceder(self, velocidad):
        self.s.send(bytes(f'{f"retroceder {velocidad}":{SIZE}}', "utf-8"))

    def izquierda(self, velocidad=10):
        self.s.send(bytes(f'{f"izquierda {velocidad}":{SIZE}}', "utf-8"))

    def derecha(self, velocidad=10):
        self.s.send(bytes(f'{f"derecha {velocidad}":{SIZE}}', "utf-8"))

    def atacar(self, velocidad):
        self.s.send(bytes(f'{f"atacar {velocidad}":{SIZE}}', "utf-8"))

    def perdonar(self):
        self.s.send(bytes(f'{"perdonar":{SIZE}}', "utf-8"))

    def apagar(self):
        self.s.send(bytes(f'{"apagar":{SIZE}}', "utf-8"))

    def detener(self):
        self.s.send(bytes(f'{"detener":{SIZE}}', "utf-8"))
```

Funciones

```python
#!/usr/bin/env python3
from ev3dev2.motor       import MoveTank, LargeMotor, MediumMotor, OUTPUT_A, OUTPUT_B, OUTPUT_C, OUTPUT_D
import socket
import threading

HOST    = ''
PORT    = 2334
SIZE    = 32

class Robot:

    def __init__(self):
        self.ruedas          = MoveTank(OUTPUT_C, OUTPUT_A)
        self.viraje          = MediumMotor(OUTPUT_D)
        self.brazo           = LargeMotor(OUTPUT_B)

        self.service         = threading.Thread(target=self.daemon, args=(), daemon=True).start()

    def avanzar(self, velocidad=10):
        self.ruedas.on(left_speed=-velocidad, right_speed=-velocidad)

    def retroceder(self, velocidad=10):
        self.ruedas.on(left_speed=velocidad, right_speed=velocidad)

    def izquierda(self, velocidad=10):
        self.viraje.on(speed=velocidad)

    def derecha(self, velocidad=10):
        self.viraje.on(speed=-velocidad)

    def atacar(self, velocidad=10):
        self.brazo.on(speed=velocidad)

    def perdonar(self):
        self.brazo.off()

    def apagar(self):
        self.ruedas.off()
        self.brazo.off()
        self.viraje.off()

    def detener(self):
        self.ruedas.off()

    def daemon(self):
        while (True):
            if(self.viraje.is_stalled): #Destraba las ruedas
                self.viraje.reset()

```

Servidor

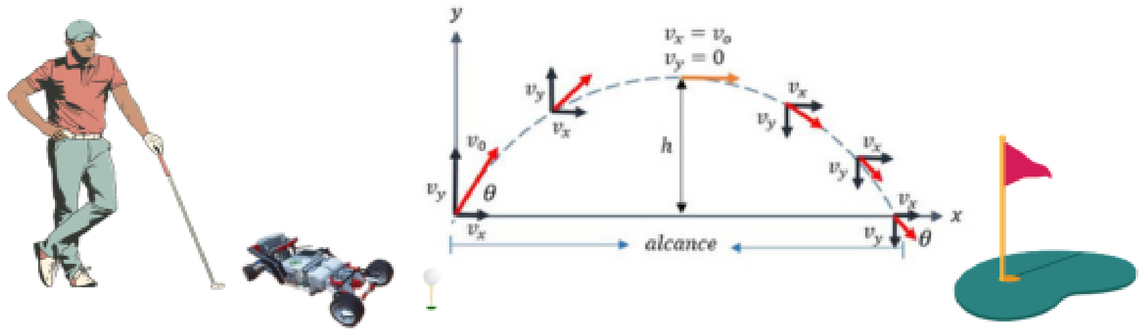
```python
if __name__ == "__main__":
    print("Inicializando servidor...")
    robot = Robot()
    exit  = False
    while True:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
            s.bind((HOST, PORT))
            s.listen(5)
            while True:
                try:
                    print("Esperando Cliente...")
                    clientsocket, address = s.accept()
                    with clientsocket:
                        print("Conectado a {}".format(address))
                        data = None
                        key  = None
                        while True:
                            data = clientsocket.recv(SIZE).decode("utf-8").split(" ")
                            key = data[0]
                            if data[1] != "":
                                value = int(data[1])

                            if key == "apagar":
                                robot.apagar()
                            elif key == "atacar":
                                robot.atacar(value)
                            elif key == "avanzar":
                                robot.avanzar(value)
                            elif key == "derecha":
                                robot.derecha(value)
                            elif key == "detener":
                                robot.detener()
                            elif key == "izquierda":
                                robot.izquierda(value)
                            elif key == "perdornar":
                                robot.perdonar()
                            elif key == "retroceder":
                                robot.retroceder(value)
                            else:
                                print("Clave desconocida: {}".format(data[0]))
                                print("Desconectando...")
                                break; #Aceptar Nuevo Cliente
                except socket.error:
                    print(str(e))
                    print("Cliente desconectado")
                except KeyboardInterrupt:
                    exit = True
                    break;
```

```python
                except Exception as e:
                    print("Error")
                    print(str(e))
                    exit = False
                    break; #Reiniciar
                robot.apagar()
        robot.apagar()
        if exit:
            print("\nSaliendo...")
            break; #Salir
        else:
            print("Reiniciando...")
```

Especificación de Requerimiento

Funcionales	No Funcionales
El robot debe poder moverse.	La interfaz gráfica debe contener los movimientos del robot (avanzar, retroceder, girar izquierda, girar derecha) además del tiro de golf.
El robot debe poder golpear una pelota hacia un punto definido.	La programación debe ser en Python.
El robot debe ser controlado por una interfaz gráfica.	La estructura del robot debe ser estable al igual que sus movimientos (construido con piezas del kit Lego Mindstorm EV3 Education).
El golpe del robot debe generar movimiento parabólico en la pelota.	El robot debe poder ser controlado desde un computador con Linux.
	La interfaz gráfica debe ser de fácil entendimiento para el usuario

Fundamentos a Proyectiles



Los datos obtenidos son:

g=9.8 m/s^2, θ= 45°, y0=0.02 m, yf= 0m, xf=1.64 m, tv= 1.13s

$$v_0 = \sqrt{\frac{g\Delta x}{\sin(2\theta)}}$$

M.R.U (Eje horizontal)

x(t)=x0+v0x∗t
1.64=0+v0*cos(45)*1.13
v0=1.64/(cos(45)*1.13)
v0=2.7627 m/s

Valor v0 para los tiros aprox ≈

M.R.U.A (Eje vertical)

y(t)=y0+v0y∗t+1/2∗g∗t^2

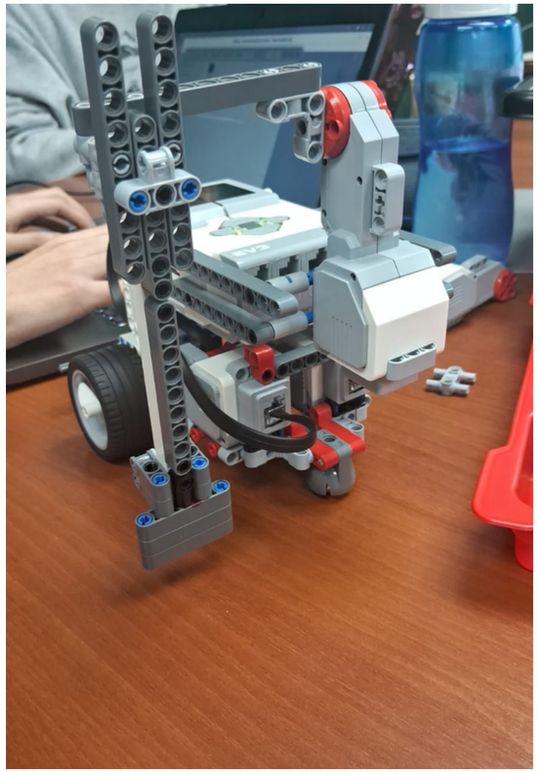
Podemos calcular la distancia final con un ángulo dado y viendo el tiempo de vuelo.

Resultados y Estado Actual

Wiki actualizada	Informe avance formulado	Conexión vía remota del robot (librería “socket”)
Carta Gantt actualizada	Bitácoras subidas (trabajo semanal)	Interfaz gráfica de movimiento (librería “tkinter”)
Funciones de movimiento del robot y tiro de golf	Estructura base de la pelota	*EXTRA* Control movimientos robot a través de mando

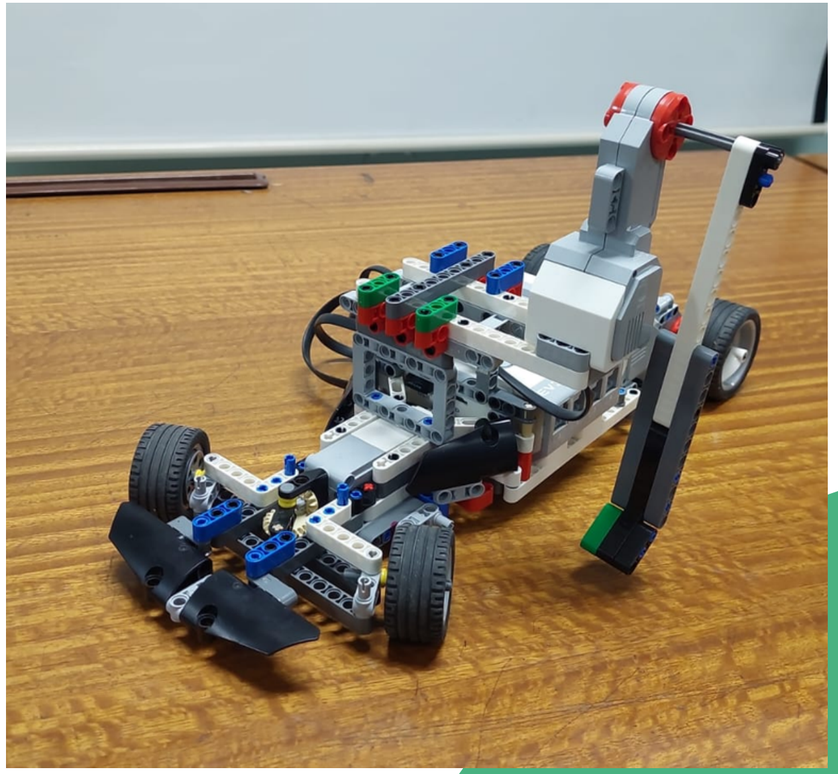
Evolución

TB1

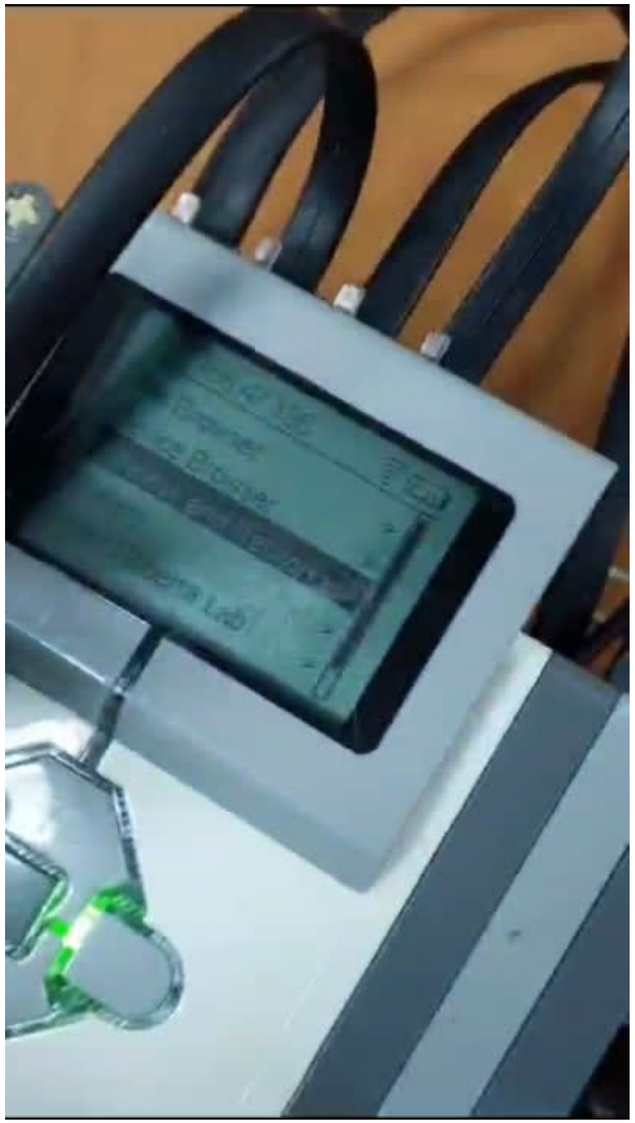


TB2

TB3



Video Prototipo



Problemas y Soluciones

Problemas	Soluciones
Tkinter entrega incorrectamente eventos de presión y soltado de tecla.	No usar el teclado para el control del robot.
Informe mal formulado	Ver la corrección del informe y arreglar errores
Dificultad para calcular el movimiento parabólico teórico	Repasar cursos pasados de mecánica clásica
Dificultad con llevar el movimiento parabólico a la práctica	Hacer una base de lego para la pelota, además de utilizar una pelota de ping pong
Calcular correctamente la estimación de costos	Hacer una separación entre costo de empleados y costo de software
Reorganizar la carta Gantt después del paro	Añadir paro a la carta Gantt y omitir semanas paradas, además de actualizar tareas y nuevas fechas

Conclusión