UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@ Computación e Informática

Informe lll
“Kiki’s Project”

Alumno(os): - Juan Casilla
- Fabián Díaz
- Andrea Navia
- Jordán Nina

Asignatura: Proyecto l

Profesor: Humberto Urrutia Lopéz

DICIEMBRE – 2023
Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
14/09/2023	1.0	Formulación del Proyecto	Juan C. Fabian D. Andrea N. Jordán N.
14/11/2023	1.1	Proyecto Avance	Juan C. Fabian D. Andrea N. Jordán N.
10/12/2022	1.4	Desarrollo de la parte final	Juan C. Fabian D. Andrea N. Jordán N.

Tabla de Contenidos

1.Panorama General

1.1.Introducción

El presente informe detalla los procedimientos empleados en el desarrollo del proyecto de construcción de un robot Lego Ev3 que simula un palo de golf. Este proyecto se desarrolló con el propósito de explorar las posibilidades de la fusión de la ingeniería mecánica y la programación para crear robots autónomos capaces de realizar acciones específicas en un contexto específico.

El objetivo principal del proyecto fue lograr que el robot se desplace de manera autónoma y que logre golpear una pelota. Para ello, se implementaron diversos algoritmos en el programa Python, que se ejecutó en el sistema Ev3Dev instalado en el robot.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar, programar y construir un robot EV3. Cuyo objetivo es simular un palo de golf, el cual es controlado a través de una interfaz gráfica del Lenguaje Python

1.2.2. Objetivos Específicos

- Realizar planificación del proyecto.
- Construir un robot con el kit EV3 Mindstorms.
- Instalar actualizaciones de software.
- Realizar un software para el movimiento del robot.
- Implementar una interfaz amigable con el usuario.

1.3. Restricciones

- El lenguaje de programación utilizado será Python.
- Un límite de tiempo para la construcción del robot y su programación.
- Todos los documentos relacionados al proyecto deberán ser subidos a la plataforma Redmine.
- El sistema operativo utilizado será Linux.

1.4. Entregable

Los archivos que se entregarán durante este proyecto son los siguientes:

- Bitácoras semanales.
- Informe de plan de proyecto.
- Presentación de plan de proyecto.
- Carta Gantt.
- Actualización del proyecto en Redmine.

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

2. Organización del Personal

2.1. Descripción de los Roles

A continuación, nombraremos los roles a cumplir y describiremos de qué tratan.

- Ensamblador: Se encarga de construir el robot, incluyendo el armado de la base, la construcción de una extremidad y el sistema de palo de golf.

- Diseñador: Se encarga de tomar fotos y videos del avance del proyecto semana tras semana y publicarlos en la wiki de Redmine.

- Documentador: Se encarga de documentar toda la información escrita del proyecto, incluyendo avances, presentaciones, bitácoras e informes.

- Jefe de grupo: Se encarga de representar al equipo y de mantener una buena organización del mismo.

- Programador: Se encarga de implementar los algoritmos necesarios para el funcionamiento del robot.

2.2. Personal que cumplirá los Roles

Las personas encargadas de cumplir los roles anteriormente mencionados son las siguientes:

Roles	Encargados
Diseñador	Andrea Navia
Programador	Fabian Diaz, Andrea Navia, Jordán Nina
Jefe de grupo	Jordán Nina
Ensamblador	Juan Casilla
Documentador	Juan Casilla , Fabian Diaz, Jordán Nina

2.3. Mecanismos de Comunicación

Los medios telemáticos que se utilizarán para la comunicación son Whatsapp y Discord, siendo este último el más importante debido a su fácil y dinámico uso, además posee la capacidad de crear canales de voz y de texto, los cuales son eficientes al momento de manejar y organizar la información que se está compartiendo entre los integrantes del grupo.

3.Planificación del Proyecto

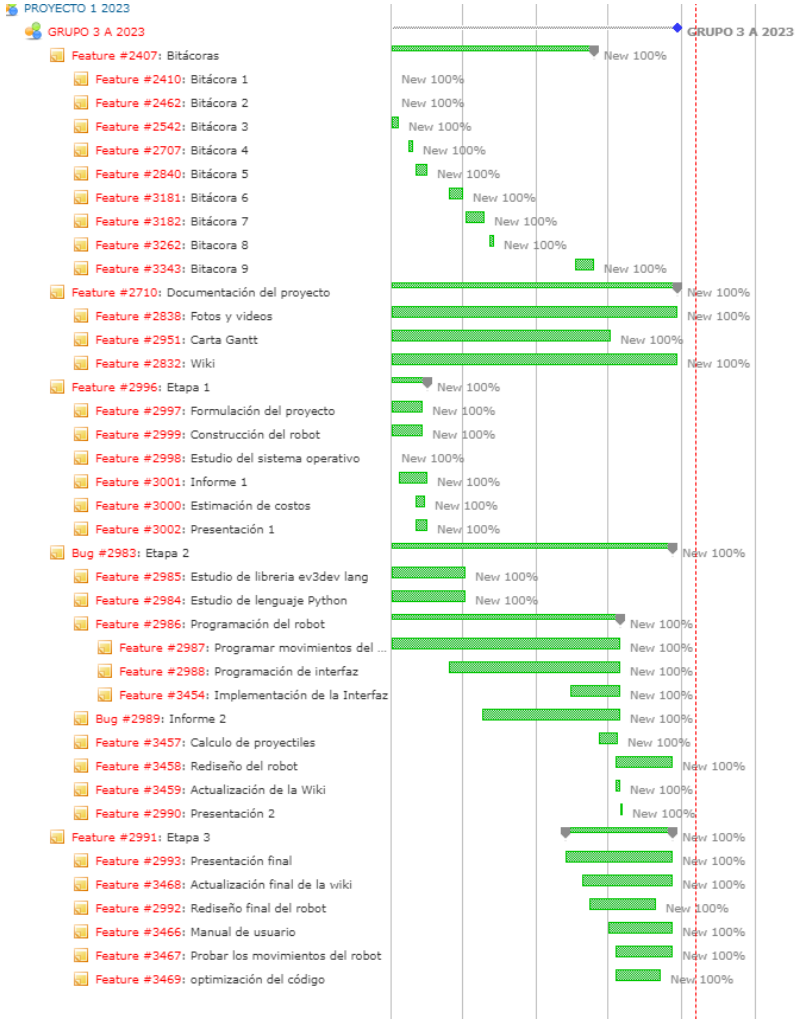
3.1.Actividades

Actividades	Descripción	Encargados
Redacción de la carta Gantt	Asignación de las tareas en relación al proyecto desde Redmine.	Juan Casilla.
Wiki	Se comparte el avance del proyecto.	Andrea Navia.
Formulación del proyecto	Análisis, Investigación, del proyecto planteado	Juan C. Fabian D. Andrea N. Jordán N.
Estudio del sistema Operativo	Análisis del funcionamiento del sistema operativo	Jordán Nina. Fabián Díaz. Andrea Navia. Juan Casilla.
Construcción del robot	Se encarga del armado	Juan Casilla.

	de la estructura y diseño del robot.	
Redacción de bitácoras	Registro de las actividades realizadas semanalmente.	Fabián Díaz.
Fotos	Encargado de sacarle fotos al robot.	Andrea Navia.
Informe I y Presentación I	Realización del primer informe y presentación.	Andrea Navia. Jordan Nina. Fabián Díaz. Juan Casilla.
Estimación de costos.	Hacer cálculo del precio total del proyecto.	Andrea Navia.
Estudio del lenguaje Python y libreria ev3dev	Investigar la programación necesaria para el proyecto	Fabian Diaz . Jordán Nina.
Programación de la interfaz	Realizar el diseño (código) de la interfaz.	Andrea Navia.
Programación de movimientos del robot	Programar e implementar movimientos del robot.	Fabian Diaz. Jordan Nina. Juan Casilla.
Implementación de la interfaz	Hacer funcional la interfaz.	Jordán Nina. Fabián Díaz.
Cálculo de proyectil	Realizar los cálculos del proyectil necesarios para lograr el objetivo.	Fabián Díaz. Juan Casilla.
Rediseño del robot	Hacer la estructura del robot más óptima tanto en estabilidad como en eficiencia a la hora del lanzamiento.	Juan Casilla. Andrea Navia. Jordan Nina.
Actualización de la Wiki	Actualizar los datos en la Wiki.	Andrea Navia.

Informe II y Presentación II	Crear y completar, informe de avance y presentación de avance.	Jordán Nina. Juan Casilla. Fabian Díaz. Andrea Navia
Presentación Final	Crear, diseñar, y desarrollar la presentación final.	Todos.
Actualización final de la Wiki	Actualizar últimos avances en la Wiki.	Andrea Navia.
Rediseño final del robot	Optimizar últimos detalles de la estructura del robot.	Juan Casilla. Jordan Nina.
Manual de usuario	Crear, diseñar, y desarrollar el manual de usuario.	Todos.
Probar los movimientos del robot	Verificar el correcto funcionamiento con respecto a la comunicación entre interfaz y servidor.	Fabian Diaz. Jordan Nina
Optimización del código	Revisar, optimizar, mejorar últimos detalles del código con respecto a la comunicación entre interfaz y servidor.	Fabian Diaz. Juan Casilla.

3.2.Asignación de Tiempo



3.3.Gestión de Riesgos

Niveles de impacto:

1. Catastrófico
2. Crítico
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 acerca del error por diversos sitios Webs y actualiza el código.
Enfermedad de algún integrante del equipo.	60%	2	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 de diseño.
Uno o más miembros dejan el proyecto.	10%	1	Reorganizar las tareas.
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%	1	Conectarlo a una fuente eléctrica y que se recargue.

4.Planificación de los Recursos

4.1.Hardware

El hardware usado en este proyecto fueron los siguientes:

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

4.2.Software

El software usado en este proyecto fueron los siguientes:

- Visual Studio Code.
- Discord.
- Linux.
- Canva.
- EV3 dev (ev3dev.org).
- Python.

4.3. Estimación de Costos

4.3.1. Costos de Hardware

Productos	Cantidad	Precio	Categoría	Total
Notebook	3	$700.000	Hardware	$2.100.000
Kit Lego MINDSTORMS (EV3)	1	$700.000	Hardware	$ 700.000
Micro SD (8 GB)	1	$5.000	Hardware	$ 5.000
Dongle USB Wifi	1	$7.000	Hardware	$ 7.000 $ 2.812.000

4.3.2. Costos de Software

Productos	Precio	Total
Visual Studio Code	x	x
Discord	x	x
Linux	x	x
Canva	x	x
EV3 dev (ev3dev.org)	x	x
WhatsApp	x	x
Python	x	Gratis

4.3.3. Costos de Gestión

Cargo	Valor hora	Horas trabajadas	Horas extras	Horas totales	Sueldo mensual	Sueldo Total
Programador	$12.000	24	8	32	$ 384.000	$ 1.920.000
Ensamblador y diseñador	$9.000	24	8	32	$ 288.000	$ 1.440.000
Jefe de Grupo	$12.000	24	6	26	$ 384.000	$ 1.920.000
Documentador	$10.000	24	2	26	$ 320.000	$ 1.600.000
Costo Total	x	x	x	x	$ 2.048.000	$ 10.240.000

4.3.4. Costos totales del proyecto

Cargo	Valor hora	Total
Costos de Hardware	$ 2.812.000	x
Costos de Software	Gratis	x
Costos de Gestión	$ 10.240.000	$13.052.000

5.Análisis y Diseño

5.1.Especificación de Requerimiento

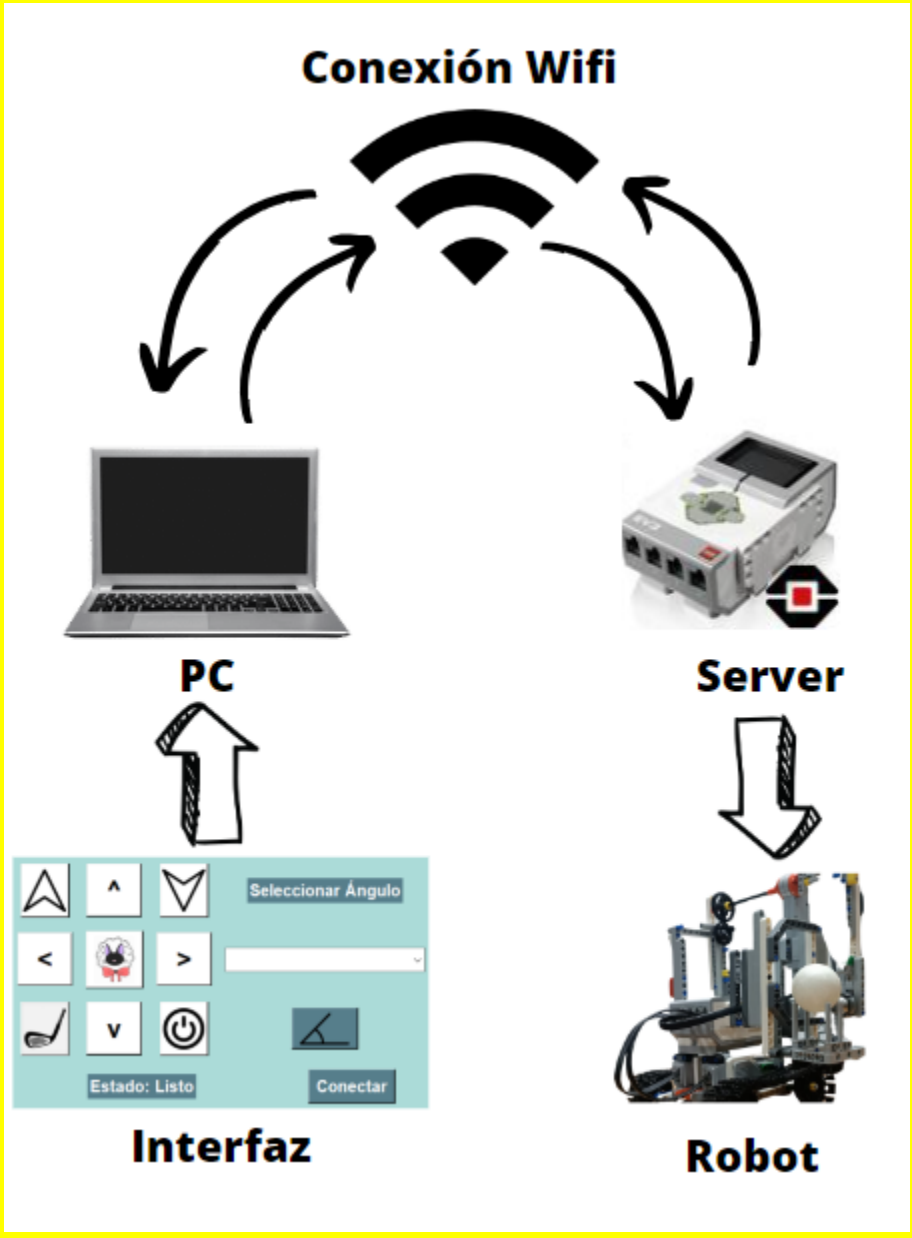
Requerimientos funcionales.

● El robot tiene la capacidad de ser controlado por una interfaz gráfica de manera remota.
● El robot está capacitado para golpear una pelota.
● El robot debe ser capaz de desplazarse lateralmente, hacia adelante y hacia atrás.
● El robot debe ser capaz de mover su palo de forma controlada.

Requerimientos no funcionales.

● Cada movimiento del robot se realizará en un tiempo y velocidad estimados.
● El diseño del robot debe ser estable y eficiente.
● El robot debe ser construido con piezas del kit Lego Mindstorm EV3 Education.
● La interfaz gráfica debe ser intuitiva y contener las funciones para que el robot sea funcional.
● El lenguaje de programación para hacer este proyecto es Python.
● El robot será manejado por un sistema operativo de Linux.

5.2. Arquitectura



● Tanto robot como computador deben estar conectados a la misma red Wifi.
● Se inicia el servidor del robot Server.py y su respectiva conexión con el computador del usuario.
● Se deberá ejecutar la interfaz gráfica para controlar el robot a distancia.
● El usuario podrá controlar el robot una vez que la interfaz gráfica se conecte al servidor del robot.
● Finalmente el robot podrá realizar los movimientos que son enviados por el usuario.

5.3.Interfaz gráfica (GUI)

En esta interfaz el usuario dispondrá de las funciones que ofrece nuestro robot.

- Botón flecha para arriba: Hace que el robot se mueva hacia adelante.
- Botón flecha a la derecha: Hace que el robot gire hacia la derecha.
- Botón flecha a la izquierda: Hace que el robot se gire hacia la izquierda.
- Botón flecha para abajo: Hace que el robot se mueva hacia abajo.
- Botón con palo de golf: Realiza el movimiento de golpear.
- Botón flecha para arriba grande : Realiza el movimiento del palo de golf hacia abajo.
- Botón flecha para arriba pequeña : Realiza el movimiento del palo de golf hacia arriba.
- Botón símbolo de apagar : Permite salir del programa en cualquier momento

6.Implementación

6.1.Fundamentos de Proyectiles

El concepto de físico que se presenta en el proyecto en desarrollo, es el lanzamiento de proyectiles . Esto ocurre cuando el palo del robot golpea la pelota con fuerza para introducir la pelota en el hoyo. Las fórmulas que explican este fenómeno se muestran a continuación.

Movimiento rectilineo uniforme (MRU): Eje horizontal

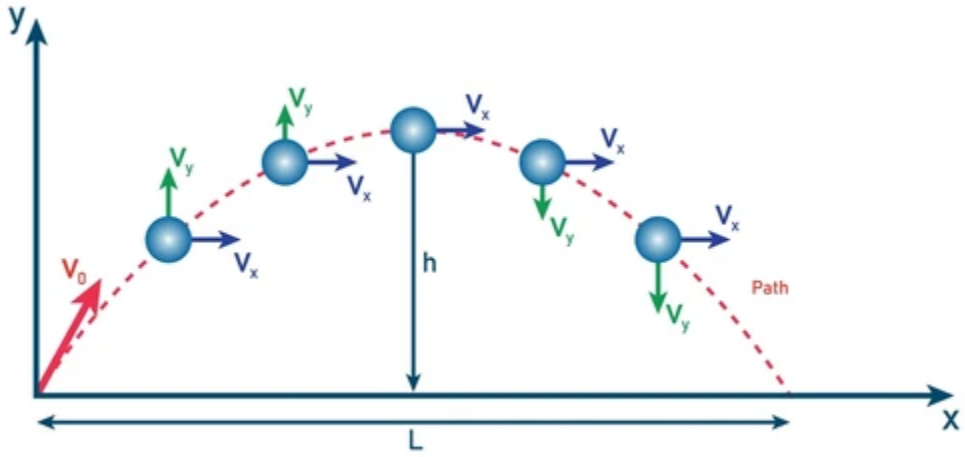
Fórmulas a utilizar:

$$x = x_0 + v_{0x} \times t$$

Movimiento rectilineo uniforme acelerado (MRUA): Eje vertical

Fórmulas a utilizar:

$$y = y_0 + v_{0y} \times t + ½ \times g \times t^2$$
$$v_y = v_0 \times \text{sen}(\theta) - g \times t$$
$$v_{ox} = v_o\cos(\theta)$$
$$v_{oy} = v_o\text{sen}(\theta)$$



Valores iniciales:

Ángulo → θ = 60°
Final (eje x)→x = 0.5 m
Posición Inicial →x_o= 0 m

Posición Final →y = 0 m
Posición. Inicial →y_o= 0.11 m

Calcularemos v_o que es la velocidad inicial con la que sale la pelota después del impacto y también calcularemos t que es el tiempo en que demora en caer la pelota después del impacto.

v_o= ?
t = ?

Primero calcularemos las velocidades iniciales en x e y.

$v_{ox}= v_o\cos(60^o)= v_o*1/2$
$v_{oy}= v_o\text{sen}(60^o)= v_o*\sqrt{3}/2$

Luego usaremos la ecuación de posición en x, despejamos v_o:

$x = x_o + v_{ox}*t \Rightarrow x = v_o/2*t \Rightarrow v_o = 2x/t$

Por consiguiente usamos la ecuacion de posicion en y:

$y = y_o + v_{oy}*t - 4.9*t^2 \Rightarrow 0 = y_o + v_o*(\sqrt{3}/2)*t - 4.9*t^2$

Reemplazando v_o y los valores iniciales en la ecuación de y:

$\Rightarrow 0 = y_o + (2x\sqrt{3}*t)/(2*t) - 4.9 * t^2$

Luego despejamos t y obtenemos el tiempo que demora en caer la pelota después del impacto.

$\Rightarrow t = \sqrt{\frac{yo + x*\sqrt{3}}{4.9}}$ = 0.44 s

Finalmente reemplazando t en la ecuación de la velocidad inicial obtenemos la velocidad con la que saldrá la pelota después del impacto v_o:

$\Rightarrow v_o = 2x/t$ = 2.27 m/s

Por lo tanto concluimos que la velocidad inicial con que inicia la pelota es 2,27 m/s. Y su tiempo de caída es 0,44(s).

6.2.Descripción de los programas

- Server

```
host = obtener_ip()
port = 12345

server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_socket.bind((host, port))

server_socket.listen(1)
print("Listening at port 12345")
print(host)

client_socket, addr = server_socket.accept()
print("Connection from: " + str(addr))
client_socket.send("Bienvenido".encode())
while True:
    data = client_socket.recv(1024).decode()
    if data=="avanzar":
        avanzar()
    elif data == "golpear":
        golpear_2()
    elif data=="atras":
        golpear_3()
    elif data=="adelante":
        golpear_1()
    elif data=="retroceder":
        retroceder()
    elif data=="derecha":
        derecha()
    elif data=="izquierda":
        izquierda()
    elif data=="desconectar":
        server_socket.close()
        print("Disconect")
    elif data=="30°":
        angulo_30()
    elif data=="45°":
        angulo_45()
```

```
    elif data=="60°":
        angulo_60()
    elif data=="90°":
        angulo_90()
```

- Funciones

```
import evdev
import ev3dev.auto as ev3
import time
import socket
from ev3dev.ev3 import *
from ev3dev2.motor import LargeMotor, OUTPUT_A, OUTPUT_B, SpeedPercent,
MoveTank
from ev3dev2.motor import LargeMotor
motorA=LargeMotor("outA")
motorD=LargeMotor("outD")
motorB=LargeMotor("outB")

## Funciones
def avanzar():
    motorA.on(-50)
    motorD.on(-50)
    time.sleep(1)
    motorA.stop()
    motorD.stop()
def golpear_1():
    motorB.on(-5)
    time.sleep(0.5)
    motorB.stop()
def golpear_2():
    mototB.on(70)
    motorB.stop()
    time.sleep(0.5)
def golpear_3():
    motorB.on(5)
    time.sleep(0.5)
    motorB.stop()
def retroceder():
    motorD.on(50,False)
```

```
    motorA.on(50,False)
    time.sleep(1)
    motorA.stop()
    motorD.stop()
def derecha():
    motorA.on(-20)
    motorD.on(20)
    time.sleep(0.5)
    motorA.stop()
    motorD.stop()
def izquierda():
    motorD.on(-20)
    motorA.on(20)
    time.sleep(0.5)
    motorD.stop()
    motorA.stop()
#funciones de inclinación del ángulo
def angulo_30():
    motorB.on_for_degrees(30,10)
def angulo_45():
    motorB.on_for_degrees(45.10)
def angulo_60():
    motorB.on_for_degrees(60,10)
def angulo_90():
    motorB.on_for_degrees(90,10)
#obtener ip de la conexión
def obtener_ip():
    try:
        # Conectarse a un servicio que devuelva la dirección IP pública.
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(("8.8.8.8", 80))  # Conectar a un servidor DNS (Google
DNS en este caso).
        ip = s.getsockname()[0]
        s.close()
        return ip
    except Exception as e:
        print(f"Error al obtener la dirección IP: {e}")
        return None
```

- Interfaz

```
import socket
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import tkinter.font as font
from tkinter import PhotoImage
import os
from tkinter import *

AVANZAR = 'avanzar'
RETROCEDER = 'retroceder'
CLOSE = 'close'
DERECHA='derecha'
IZQUIERDA='izquierda'
GOLPEAR='golpear'
ATRAS='atras'
ADELANTE='adelante'
DESCONECTAR='desconectar'
ANGULO_30='30°'
ANGULO_45='45°'
ANGULO_60='60°'
ANGULO_90='90°'
MOVER='mover'

def conectar(ventana):
    try:
        port=12345
        host=direccion_ip.get()
        global client
        client = socket.socket()
        client.connect((host, port))
        print(client.recv(1024).decode())
        estado_label.config(text="Estado: Conectado")
        boton_conectar.config(state=tk.DISABLED)
        ventana.destroy()
    except socket.error:
        estado_label.config(text="Estado: Fallo al conectar")
        boton_conectar.config(state=tk.NORMAL)
        ventana.destroy()
```

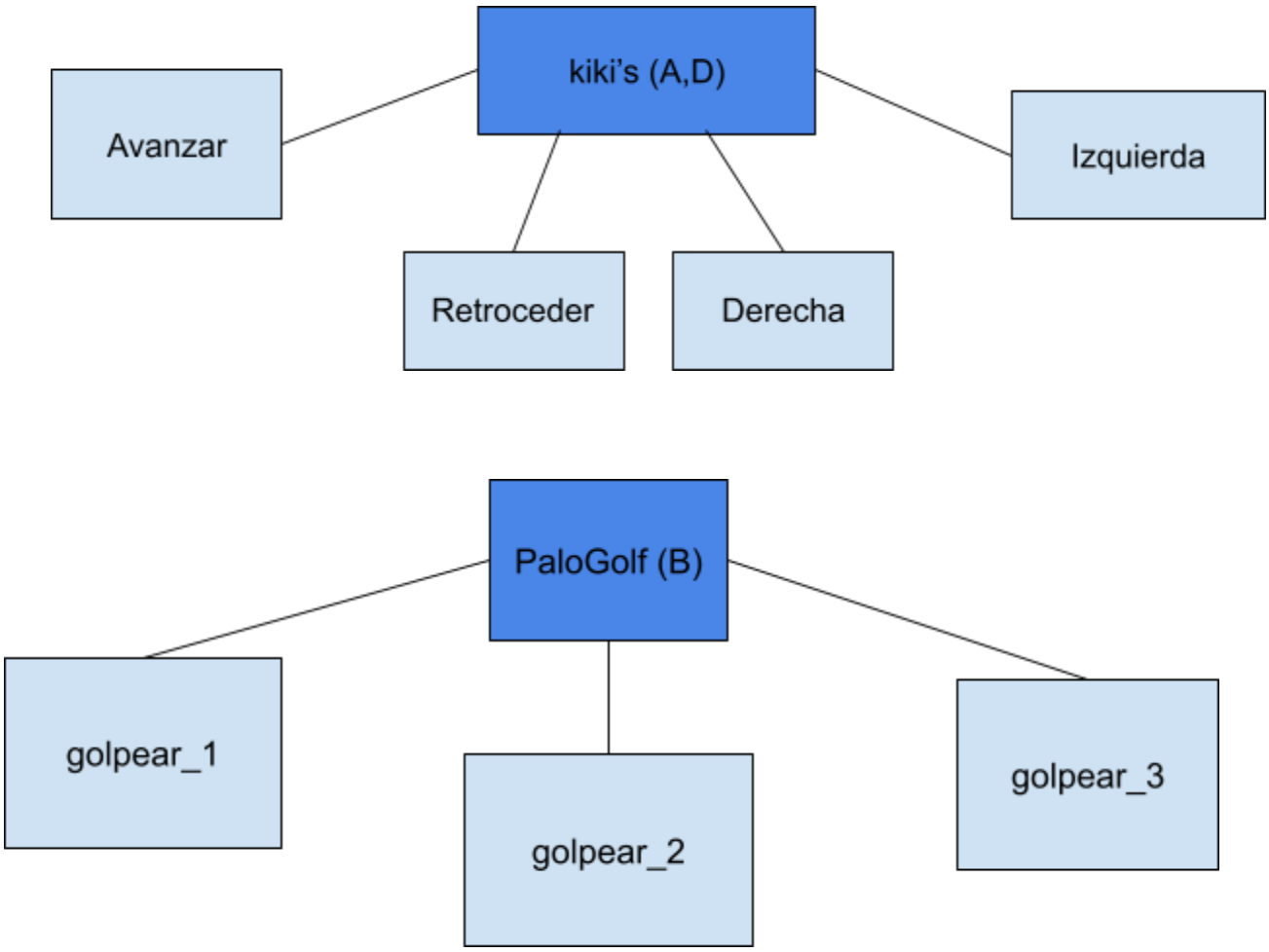
```
        messagebox.showwarning("Conexión errónea","No se ha logrado la
conexión, verifique la IP")

def ventana_ip():
    boton_conectar.config(state=tk.DISABLED)
    ventana_2 = tk.Toplevel(ventana)
    ventana_2.title("Conexión a servidor")
    ventana_2.configure(bg="#ABDBD8")
    ventana_2.geometry("150x120")

    etiqueta_ip = tk.Label(ventana_2, fg=a,text="Ingresar dirección de
IP:", bg="#567E8B")
    etiqueta_ip.pack(pady=10)

    global direccion_ip
    direccion_ip= tk.Entry(ventana_2,bg=a)
    direccion_ip.pack(pady=10)
    estado_label.config(text="Estado: Conectando")

    boton_ingresar = tk.Button(ventana_2, text="Ingresar",fg=a,
command=lambda:conectar(ventana_2),bg="#567E8B")
    boton_ingresar.pack(pady=10)

# funciones
def avanzar():
    client.send(AVANZAR.encode())
    estado_label.config(text="Estado: Avanzando")
    print("Avanzando")
def retroceder():
    client.send(RETROCEDER.encode())
    print("Retrocediendo")
    estado_label.config(text="Estado: Retrocediendo")
def girar_der():
    client.send(DERECHA.encode())
    estado_label.config(text="Estado: Girando hacia la Derecha")
    print("Girando derecha")
def girar_izq():
    client.send(IZQUIERDA.encode())
    estado_label.config(text="Estado: Girando hacia la Izquierda")
    print("Girando izquierda")
```

```
def rotar_brazo_Atras():
    client.send(ATRAS.encode())
    estado_label.config(text="Estado: Ajustando brazo hacia Atras")
def rotar_brazo_Adelante():
    client.send(ADELANTE.encode())
    estado_label.config(text="Estado: Ajustando brazo hacia Adelante")
def angulo():
    angulo_seleccinado=etiquetas_angulo.get()
    if(angulo_seleccinado=="30°"):
        etiquetas_angulo.set('')
        client.send(ANGULO_30.encode())
        estado_label.config(text="Estado: Inclinado ángulo 30°")
    elif(angulo_seleccinado=="45°"):
        etiquetas_angulo.set('')
        client.send(ANGULO_45.encode())
        estado_label.config(text="Estado: Inclinado ángulo 45°")
    elif(angulo_seleccinado=="60°"):
        etiquetas_angulo.set('')
        client.send(ANGULO_60.encode())
        estado_label.config(text="Estado: Inclinado angulo 60°")
    elif(angulo_seleccinado=="90°"):
        etiquetas_angulo.set('')
        client.send(ANGULO_90.encode())
        estado_label.config(text="Estado: Inclinado ángulo 90°")
    else:
        estado_label.config(text="Estado: Ángulo no encontrado")

def golpear():
    client.send(GOLPEAR.encode())
    estado_label.config(text="Estado: Golpeando")
def desconectar():
    client.send(DESCONECTAR.encode())
    estado_label.config(text="Estado: Desconectado")
    client.close()
    ventana.destroy()
def mover():
    client.send(MOVER.encode())
    estado_label.config(text="Estado: Realizando Movimientos")
# Crear la ventana principal de Tkinter
ventana = tk.Tk()
```

```
ventana.title("Control de Robot con Palo de Golf")
ventana.config(bg="#ABDBD8")
a="white"
buttonFont = font.Font(family='Helvetica', size=24, weight='bold')
buttonFontSec = font.Font(family='Helvetica', size=15, weight='bold')
boton_avanzar = tk.Button(ventana, text="^", command=avanzar, height=1,
width=3, font=buttonFont,bg=a)
boton_retroceder = tk.Button(ventana, text="v", command=retroceder,
height=1, width=3, font=buttonFont,bg=a)
boton_izquierda = tk.Button(ventana, text="<", command=girar_izq,
height=1, width=3, font=buttonFont,bg=a)
boton_derecha = tk.Button(ventana, text=">", command=girar_der, height=1,
width=3, font=buttonFont,bg=a)

imagen_abajo=PhotoImage(file="C:/Users/nombre/Desktop/abajo.png")
boton_rotar_atras = tk.Button(ventana,image=imagen_abajo,
command=rotar_brazo_Atras, height=60, width=53, font=buttonFontSec,bg=a)

imagen_arriba=PhotoImage(file="C:/Users/nombre/Desktop/arriba.png")
boton_rotar_adelante = tk.Button(ventana,image=imagen_arriba,
command=rotar_brazo_Adelante, height=60, width=53,
font=buttonFontSec,bg=a)

imagen_golpe=PhotoImage(file="C:/Users/nombre/Desktop/golpe.png")
boton_golpear = tk.Button(ventana,image=imagen_golpe, command=golpear,
height=60, width=53, font=buttonFontSec)
imagen_angulo=PhotoImage(file="C:/Users/nombre/Desktop/angulo.png")
boton_angulo= tk.Button(ventana, image=imagen_angulo, command=angulo,
height=45 ,width=75,fg="white" ,font=buttonFontSec,bg="#567E8B")

imagen_apagar=PhotoImage(file="C:/Users/nombre/Desktop/apagar.png")
boton_apagar = tk.Button(ventana, image=imagen_apagar,
command=desconectar, height=60, width=53, font=buttonFontSec,bg=a)

boton_conectar = tk.Button(ventana, text="Conectar", command=ventana_ip,
height=1, width=8,fg="white" ,font=buttonFontSec,bg="#567E8B")
boton_conectar.grid(column=4, row=6, padx=10, pady=8, rowspan=1)

# Agrega la etiqueta para "Seleccionar Ángulo" y la caja desplegable
```

```
etiqueta_angulo = tk.Label(ventana, text="Seleccionar Ángulo", fg="white"
,  font=buttonFontSec,bg="#567E8B")
etiquetas_angulo = ttk.Combobox(ventana, values=["30°", "45°",
"60°","90°"], font=buttonFontSec)

etiqueta_angulo.grid(column=3, row=0, padx=8, pady=8, columnspan=2,
rowspan=1)
etiquetas_angulo.grid(column=3, row=1, padx=8, pady=8, columnspan=2,
rowspan=1)
botón angulo.grid(column=3, row=2, padx=8, pady=8, columnspan=2,
rowspan=1)


imagen = PhotoImage(file="C:/Users/nombre/Desktop/KIKIS.png")
logo = tk.Button(ventana, image=imagen,command=mover, height=64, width=64,
font=buttonFontSec,bg=a)
boton_rotar_adelante.grid(column=0, row=0, padx=8, pady=8)
boton_avanzar.grid(column=1, row=0, padx=8, pady=8)
boton_rotar_atras.grid(column=2, row=0, padx=8, pady=8)

boton_izquierda.grid(column=0, row=1, padx=8, pady=8)
logo.grid(column=1, row=1, padx=8, pady=8)
boton_derecha.grid(column=2, row=1, padx=8, pady=8)

boton_golpear.grid(column=0, row=2, padx=8, pady=8)
boton_retroceder.grid(column=1, row=2, padx=8, pady=8)

# Agrega el botón "Mover" debajo del botón de la flecha hacia la derecha
boton_apagar.grid(column=2, row=2, padx=8, pady=8)
estado_label = tk.Label(ventana, text="Estado: Desconectado",  fg="white"
,  font=buttonFontSec,bg="#567E8B")
estado_label.grid(row=6, column=0, columnspan=4, pady=8)
ventana.mainloop()
```

6.3. Diagrama



7.Resultados

7.1.Estado Actual del Proyecto

A continuación un video del robot ya terminado:

Link: http://pomerape.uta.cl/redmine/documents/1570

7.2.Problemas Encontrados y Solución Propuesta

Problemas encontrados	Soluciones
Problemas en la conexión de la interfaz con el servidor, esto surge por una red wifi lenta o con mala cobertura.	Para probar el robot, se recomienda hacerlo en momentos en los que no haya otros grupos conectados a la misma red inalámbrica. Si esto no es posible, compartir datos móviles del celular de uno de los integrantes.
Fallas a la hora de golpear, el brazo del robot a la hora de golpear una pelota no se posiciona de manera correcta ,provocando tiros en posiciones aleatorias.	Seleccionar un ángulo de golpeo, para que este realice un golpe en una posición determinada.
Una estructura muy débil y poco sólida, esto hace que el robot se desarme mientras se ejecuta los movimientos.	Rediseño de la estructura del robot ,agregando un contra peso en el lado opuesto
Fallas del robot a la hora de girar hacia los lados, esto provoca saltos innecesarios a la hora de girar, pudiendo provocar el desgaste del motor.	Usar 2 motores, uno que avanza hacia adelante y otro hacia atrás, para que este pueda girar sin complicaciones.
El robot se descarga o apaga durante el desarrollo del mismo.	Cargar el robot antes de encenderlo y usarlo cada cierto lapso de tiempo porque si pasa mucho tiempo sin usarlo, se apaga aunque tenga batería.

Falta de creatividad a la hora de desarrollar el código, esto impide el desarrollo del código.	Lluvia de ideas, ver diversidad de opiniones, plasmar el problema en una hoja de papel y desencriptar el mismo.
Falta de creatividad a la hora de diseñar la estructura del robot, esto impide el desarrollo del mismo.	Ver posibles soluciones, escuchar diversidad de opiniones, plasmar el problema en una hoja de papel y desencriptar el mismo.

8.Conclusión

8.1.Conclusiones Generales

Una vez finalizado el proyecto del robot, podemos decir que cumplimos con lo solicitado al principio de todo esto- El robot, controlado a través de una interfaz, es capaz de moverse y golpear una pelota. Podríamos haber agregado más funciones o mejorar algunos aspectos del robot, pero en general estamos satisfechos con el trabajo realizado y con el resultado final del robot. Mejoramos nuestra capacidad de autoaprendizaje y salimos fortalecidos superando las dificultades que se presentaron a lo largo de todo este tiempo.

8.2.Trabajo futuro

Como grupo, estamos muy satisfechos con los resultados obtenidos. El robot cumple con todos los requisitos especificados por el cliente, es decir, es capaz de simular un golpe de golf con precisión, tiene una estructura sólida y es controlado por una interfaz gráfica.

Sin embargo, siempre estamos buscando formas de mejorar nuestros productos, y el robot de golf no es una excepción. Para un futuro, nos gustaría implementar las siguientes mejoras:

Aumento de la precisión: sensores más precisos o algoritmos de control más sofisticados.

Aumento de la potencia: motores más potentes u optimización de la trayectoria del golpe.

Mejora de la facilidad de uso: interfaz más intuitiva o simplificación de los procedimientos de configuración.

Creemos que estas mejoras harán que el robot de golf sea una herramienta aún más óptima y sencilla de usar.

9. Referencias (IEEE)

Ev3dev home. (s. f.). ev3dev. https://www.ev3dev.org/

Ev3dev. (s. f.). *GitHub - ev3dev/ev3dev-lang-Python: Pure Python bindings for EV3Dev*. GitHub. https://github.com/ev3dev/ev3dev-lang-python

Nigel Ward. (2016, 27 octubre). *EV3 Python: Set up an SSH connection from the EV3 to the computer* [Vídeo]. YouTube. https://www.youtube.com/watch?v=ZfhqZGFJd9A

8th Man Robotics. (2019, 23 septiembre). *Simple tracked vehicle build instructions for LeGO Mindstorms EV3 Education Core Set* [Vídeo]. YouTube. https://www.youtube.com/watch?v=c28Qi1Fg64o

LEGORobotics Mr. Hino. (2020, 25 junio). *«My LEGO EV3 mini Golf Robot!!» trick shot too!!!* [Vídeo]. YouTube. https://www.youtube.com/watch?v=lq3YhkSH3Sg

Guard Tank - simple LEGO Mindstorms robot with treads. (s. f.). FLLCasts. https://www.fllcasts.com/materials/369-guard-tank-simple-lego-mindstorms-robot-with-treads#is-js-view