UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@
Computación e Informática

Plan de Proyecto
“Kik9s Project”

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

Asignatura: Proyecto I

Profesor: Humberto Urrutia

Noviembre 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.

Tabla de Contenidos

1. Panorama General 4
    1.1. Introducción 4
    1.2. Objetivos 4
        1.2.1. Objetivo General 4
        1.2.2. Objetivo Específico 4
    1.3. Restricciones 4
    1.4. Entregables 5
2. Organización del Personal 5
    2.1. Descripción de los Roles 5
    2.2. Personal que cumplirán los Roles 6
    2.3. Mecanismos de Comunicación 6
3. Planificación del Proyecto 6
    3.1. Actividades 6
    3.2. Asignación de Tiempo 9
    3.3. Gestión de Riesgos 10
4. Planificación de los Recursos 11
    4.1. Hardware 11
    4.2. Software 11
    4.3. Estimación de Costos 11
5. Análisis y Diseño 13
    5.1. Especificación de Requerimiento 13
    5.3. Arquitectura 14
    5.4. Interfaz 15
6. Implementación 16
    6.1. Fundamentos de Proyectiles 16
    6.2. Descripción de los programas 17
    6.3. Diagramas 22
7. Resultados 22
    7.1. Estado Actual del Proyecto 23
    7.2. Problemas Encontrados y Solución Propuesta 23
8. Conclusión 24
9. Referencias 24

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.

El informe está estructurado de la siguiente manera:

Organización del personal: Se detalla la división de roles y los canales de comunicación.
Planificación del proyecto**:** Se evalúan los temas relacionados a la distribución del tiempo.
Planificación de recursos: Se detallan los hardware y software usados y los respectivos costos.

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. Objetivo Específico

- 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. Entregables

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 cumplirán 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 de la estructura y diseño del robot.	Juan Casilla.
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.

3.2. Asignación de Tiempo

- Planificación de proyecto: 3-4 semanas.
- Ejecución del proyecto: 5-6 semanas.
- Finalización del proyecto: 16 semanas.

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)
- WhatsApp
- Python

4.3. Estimación de Costos

Productos	Cantidad	Precio	Categoría	TOTAL
Notebook	3 unidades	$700.000 c/u	Hardware	$ 2.100.000
Kit Lego MINDSTORMS (EV3)	1 unidad	$700.000	Hardware	$ 700.000
Micro SD (8 GB)	1 unidad	$5.000	Hardware	$ 5.000
Dongle USB Wifi	1 unidad	$7.000	Hardware	$ 7.000 $ 2.812.000

Cargo	Valor horas trabajadas	Horas trabajadas	Horas extras trabajadas	Horas totales trabajadas	Horas totales trabajadas	Sueldo mensual	Sueldo TOTAL
PROGRAMADOR	12.000	24	8	32	32	$ 384.000	$ 1.920.000
ENSAMBLADOR Y DISEÑADOR	9.000	24	8	32	32	$ 288.000	$ 1.440.000
JEFE DE GRUPO	12.000	24	6	26	26	$ 384.000	$ 1.920.000
DOCUMENTADOR	10.000	24	2	26	26	$ 320.000	$ 1.600.000
COSTO TOTAL	X	X	X	X	X	$ 2.048.000	**$ 10.240.000**

5. Análisis y Diseño

5.1. Especificación de Requerimiento

5.2.

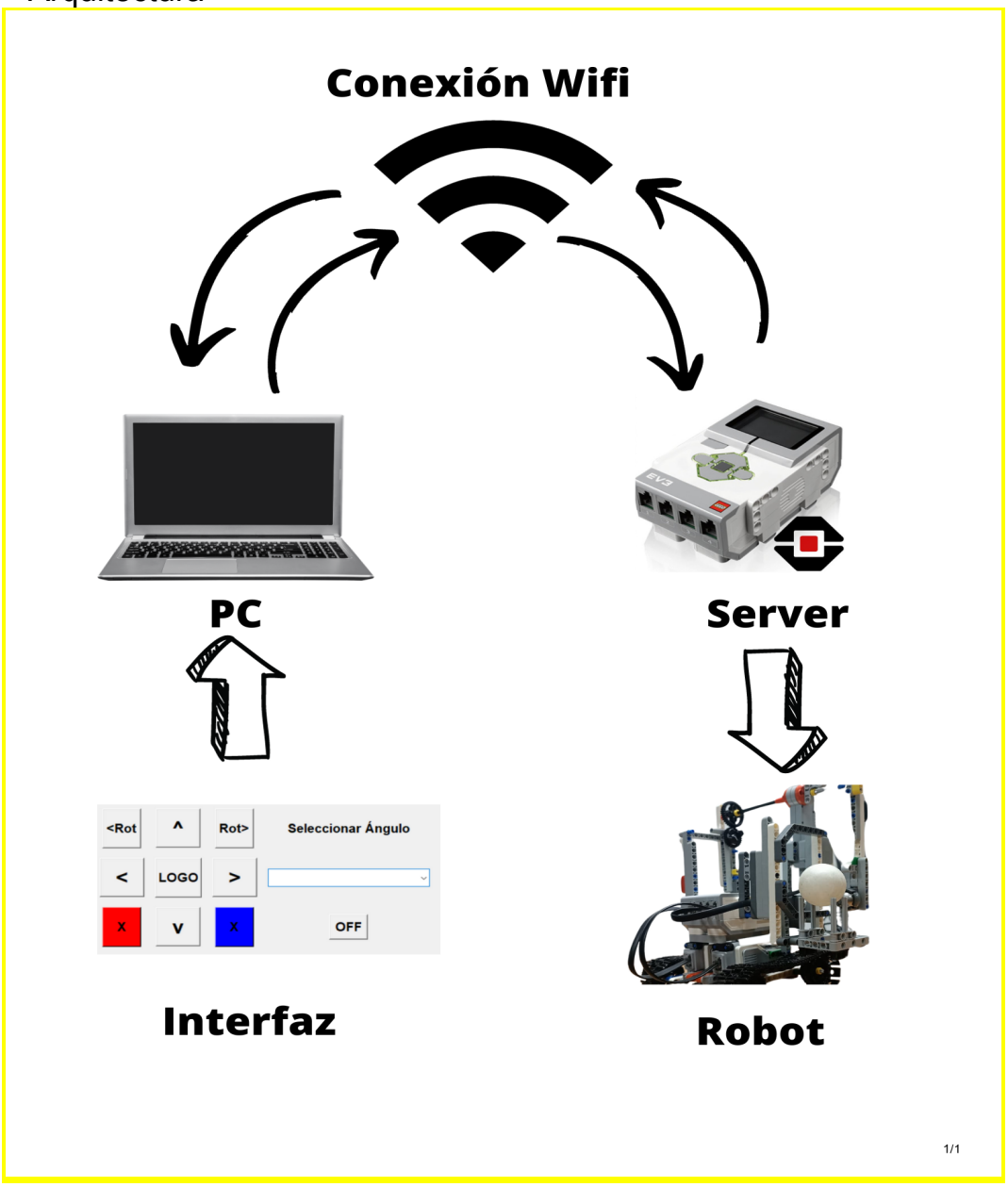
Requerimientos funcionales.

- 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.

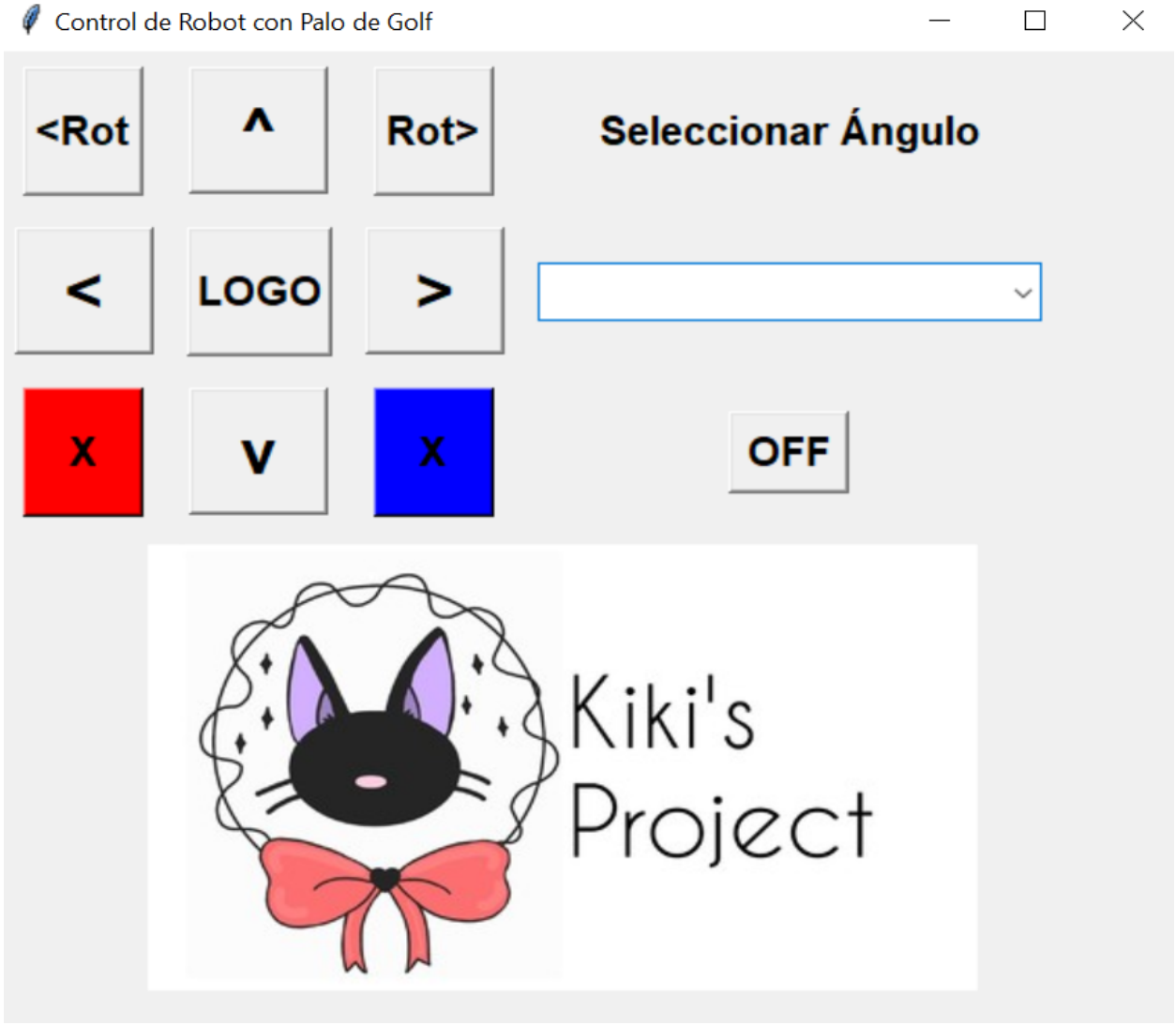
- Cada movimiento del robot se realizará en un tiempo y velocidad estimados.
- El diseño del robot debe ser estable y eficiente.
- El robot debe ser construido con piezas del kit Lego Mindstorm EV3 Education.
- La interfaz gráfica debe ser intuitiva y contener las funciones para que el robot sea funcional.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

5.3. Arquitectura



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

5.4. Interfaz



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 una X roja : Realiza el movimiento de golpear.
- Botón Rot> : Realiza el movimiento del palo de golf hacia atrás.
- Botón Rot> : Realiza el movimiento del palo de golf hacia adelante.

6. Implementación

6.1. Fundamentos de Proyectiles

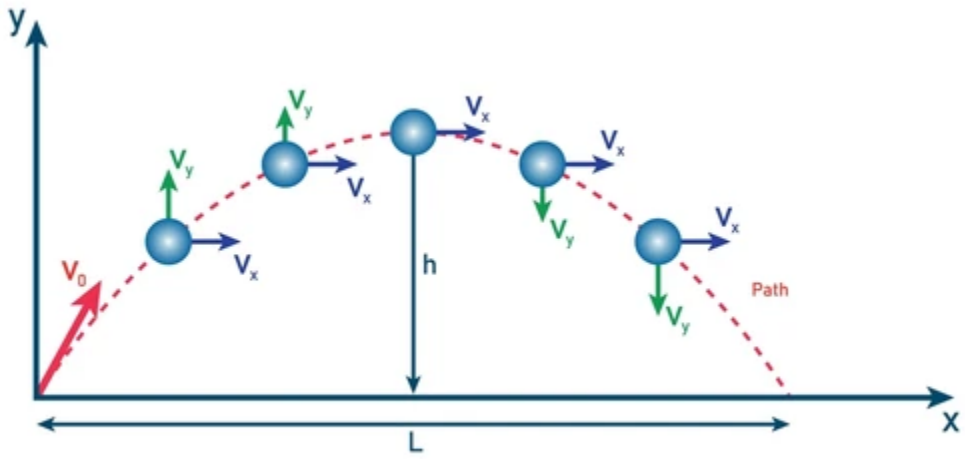
El concepto de físico que se presenta en el proyecto en desarrollo, es el lanzamiento de proyectiles . Esto ocurre cuando el palo del robot golpea la pelota con fuerza para introducir la pelota en el hoyo. Las fórmulas que explican este fenómeno se muestran a continuación.

Movimiento rectilineo uniforme (MRU): Eje horizontal

$$x = x_0 + v_{0x} \times t$$
$$v_x = v_0 \times \cos(\theta)$$

Movimiento rectilineo uniforme acelerado (MRUA): Eje vertical

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



$\theta = 60^o$
$x = 0.5$ m
$x_o = 0$ m
$y = 0$ m
$y_o = 0.11$ m
$v_o = ?$
$t = ?$

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

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

reemplazando v_o en la ecuación de y:

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

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

reemplazando t en 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:

```
#Se define el puerto y la IP”
host = "192.168.71.1"
port = 12345

#Se crea un nuevo objeto de socket para el servidor.
server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_socket.bind((host, port))#enlazamos el puerto y host al server

server_socket.listen(1)#permite que el servidor escuche una conexión
print("Listening at port 12345")

client_socket, addr = server_socket.accept()#Esta línea bloquea la
ejecución del código hasta que se establezca una conexión con el servidor.
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()
```

- 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(-10)
    time.sleep(0.5)
    motorB.stop()
def golpear_2():
    motorB.on(-70)
```

```
    time.sleep(0.1)
    motorB.stop()
def golpear 3():
    motorB.on(10)
    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()
```

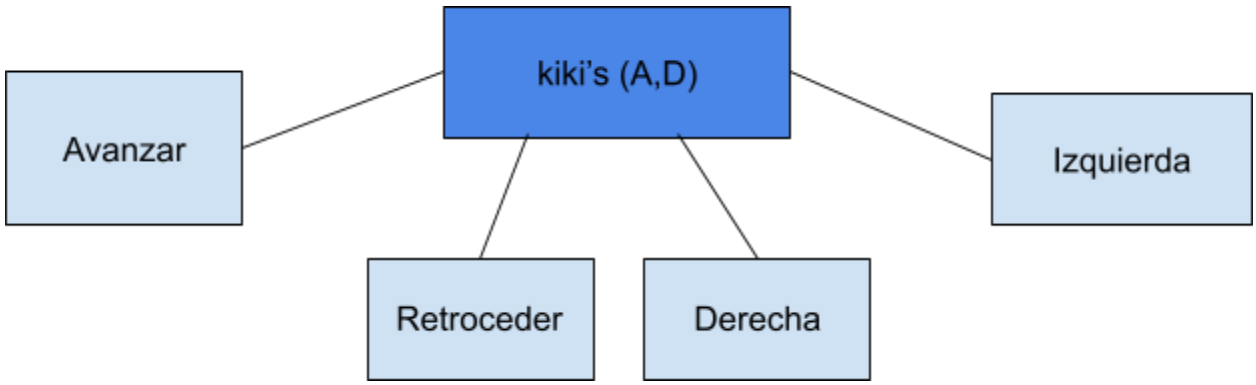
- Cliente

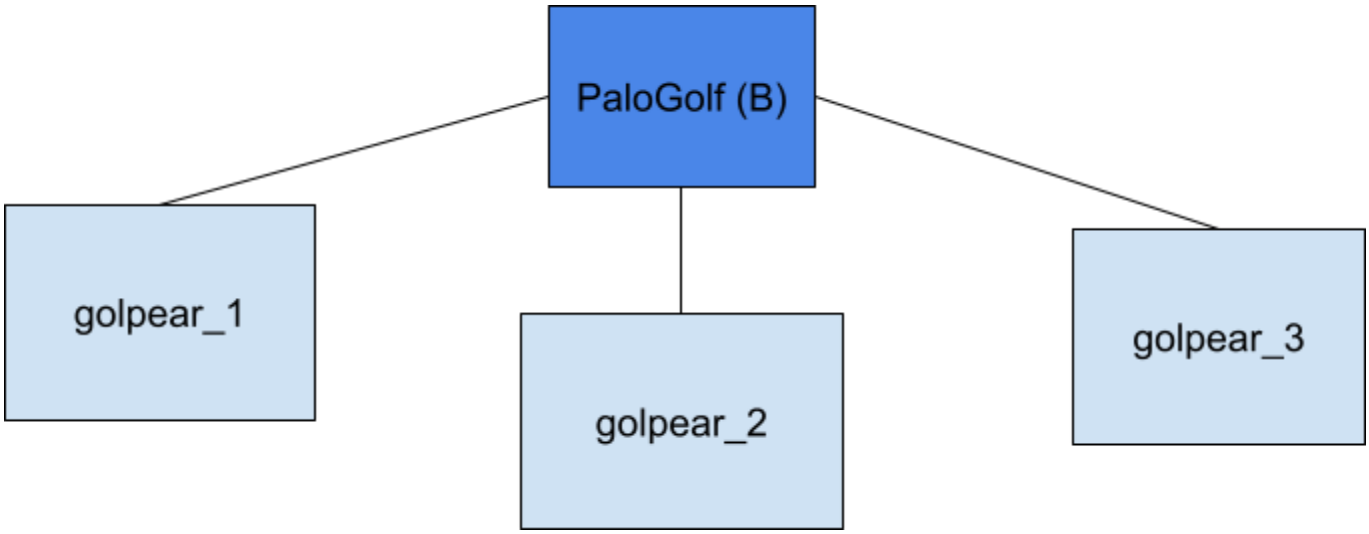
```
import tkinter as tk
import tkinter.font as font
import socket

PORT = 12345
SERVER = '192.168.71.1'

AVANZAR = 'avanzar'
RETROCEDER = 'retroceder'
DETENER = 'detener'
CLOSE = 'close'
DERECHA='derecha'
IZQUIERDA='izquierda'
GOLPEAR='golpear'
ATRAS='atras'
```

```
ADELANTE='adelante'

if __name__ == '__main__':
    client = socket.socket()
    client.connect((SERVER, PORT))
    print(client.recv(1024).decode())
    def avanzar():
        client.send(AVANZAR.encode())
        print("Avanzando")
    def retroceder():
        client.send(RETROCEDER.encode())
        print("Retrocediendo")
    def girar_der():
        client.send(DERECHA.encode())
        print("Girando derecha")
    def girar_izq():
        client.send(IZQUIERDA.encode())
        print("Girando izquierda")
    def rotar_brazo_Atras():
        client.send(ATRAS.encode())
        print("Rotando brazo hacia la derecha")
    def rotar_brazo_Adelante():
        client.send(ADELANTE.encode())
        print("rotando brazo hacia la izquierda")
    def parar_robot():
        client.send(DETENER.encode())
        print("Se detuvo el robot")
    def golpear():
        client.send(GOLPEAR.encode())
        print("golpeando...")
    ventana = tk.Tk()
    ventana.title("Control de Robot con Palo de Golf")
    buttonFont = font.Font(family='Helvetica', size=24, weight='bold')
    buttonFontSec = font.Font(family='Helvetica', size=15, weight='bold')

    botón avanzar = tk.Button(ventana, text="^", command=avanzar,
height=1, width=3, font=buttonFont)
    boton_retroceder = tk.Button(ventana, text="v", command=retroceder,
height=1, width=3, font=buttonFont)
```

```
    boton_izquierda = tk.Button(ventana, text="<", command=girar_izq,
height=1, width=3, font=buttonFont)
    boton_derecha = tk.Button(ventana, text=">", command=girar_der,
height=1, width=3, font=buttonFont)

    boton_rotar_atras = tk.Button(ventana, text="Rot>",
command=rotar_brazo_Atras, height=2, width=4, font=buttonFontSec)
    boton_rotar_adelante = tk.Button(ventana, text="<Rot",
command=rotar_brazo_Adelante, height=2, width=4, font=buttonFontSec)

    boton_stop = tk.Button(ventana, text="X", command=golpear, height=2,
width=4, font=buttonFontSec, bg='red')
    boton_apagar = tk.Button(ventana, text="OFF", command=parar_robot,
height=2, width=6, font=buttonFontSec)
    logo = tk.Button(ventana, text= "LOGO", height=2, width=5,
font=buttonFontSec)

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

    ventana.mainloop()
    client.close()
```

6.3. Diagramas





7. Resultados

7.1. Estado Actual del Proyecto

Actualmente el robot cumple con todas las funcionalidades descritas con anterioridad. El ensamblaje básico está igualmente implementado, por lo que se ha cumplido con lo propuesto en la fase de formulación del proyecto. Por el lado de la programación se ha cumplido con lo propuesto. Finalmente, en la plataforma Redmine, se ha ido actualizando los avances grupales en la carta Gantt, faltando solamente actualizar la Wiki, la cual será más tomada en cuenta en la fase final del proyecto.

7.2. Problemas Encontrados y Solución Propuesta

Problemas encontrados	Soluciones
Fallas en la comunicación con el robot debido a una conexión a internet débil en ciertas circunstancias	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, se debe generar una conexión con un dispositivo móvil.
Fallas a la hora de golpear , la estructura para 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.
Inestabilidad a la hora de golpear la pelota debido al diseño mal balanceado el robot se balancea hacia un lado en específico	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, provocando 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.

8. Conclusión

Durante el desarrollo de este proyecto se pudieron apreciar dificultades, principalmente en el diseño final del robot, debido a que antes de llegar a construir “kiki 's project” se armó un modelo distinto estéticamente. No obstante actualmente se cuenta con un diseño estable, en el cual se pueda programar los movimientos del robot para que este tenga la capacidad de moverse en distintas direcciones controladas por el ususario.

Por otro lado, debido a todos los inconvenientes que se han presentado, como la salida de uno de los integrantes de equipo, se han reasignado las tareas a cada uno de los integrantes para poder sobrellevar la situación y reforzar la organización del equipo.

9. Referencias

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