UNIVERSIDAD DE TARAPACÁ

UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Plan de Proyecto
“ANT-0T0”

Alumno(os): Diego Ferrada
Fabian Quezada
Javier Huanca
Joshua Jara
Maykol Bravo

Asignatura: Proyecto I

Profesor: Humberto Urrutia

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
01/09/2023	1.0	Inicio de la redacción del informe de formulación del proyecto	Diego Ferrada Javier Huanca
11/09/2023	1.1	Cambios en el Panorama General. En específico en el ítem 1.1 y 1.2 Modificaciones en el ítem 2.1 y 2.2 Corrección en porcentajes ítem 3.3	Fabian Quezada
17/11/2023	1.2	Corrección al índice 1.1, se replantea el panorama general Correcciones al índice 1.2, se expanden los objetivos específicos y se replantea el objetivo general Actualización a la asignación de roles Corrección de la estimación de costos	Joshua Jara Fabian Quezada

27/11/2023	1.3	Se añaden los índices correspondientes a la fase 2 Actualización a la estimación de costos	Joshua Jara Fabian Quezada

Tabla de contenidos

1. Panorama General

1.1 Introducción

El minigolf es una variante del golf tradicional; la distinción principal radica en que se juega en dimensiones considerablemente más reducidas, tanto en términos de área como de extensión del campo de golf. Aunque generalmente sigue las mismas reglas, el minigolf presenta algunas variaciones en la ejecución y no requiere conocimientos previos de golf ni una condición física excepcional para lograrlo.

Considerando los requerimientos fácilmente adquiribles del minigolf, se considera plausible la idea de crear un robot capaz de conseguir el mismo objetivo ¿Qué herramientas podrían utilizarse para lograr construir un robot con estas capacidades?.

Lego Mindstorms constituye una serie de robótica dirigida a niños originalmente introducida en 1998, desarrollada por la compañía LEGO. Esta línea incorpora principios fundamentales de la robótica, como la ensambladura de piezas y la programación de acciones de manera interactiva. Gracias al modelo inicial y los kits posteriores de Lego Mindstorms es posible ensamblar incluso robots a gran escala, por lo tanto, un robot que juegue golf cae dentro de este horizonte de posibilidades.

En esta asignatura, se planificará un proyecto que cumpla con el objetivo de ensamblar dicho robot utilizando los conocimientos adquiridos en los últimos dos años de carrera además de el Kit Lego Mindstorms EV3. Se plantea primero la planificación de tareas y roles. Luego, se examinan los posibles riesgos durante la fase de ejecución, además de considerar cuáles serán los recursos necesarios para elaborar un presupuesto detallado.

Además, en la segunda fase del proyecto, se detalla el proceso de ensamblado, aprendizaje de la biblioteca ev3dev2 y testeo de código del robot. Primero, se adjuntan imágenes y detalles de las distintas iteraciones del robot. Luego, se define el método de aprendizaje de la biblioteca mencionada y se describe el código funcional y no funcional del proyecto.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar un robot utilizando el kit LEGO Mindstorms EV3 con la capacidad de realizar lanzamientos precisos mediante la utilización de fórmulas previamente estudiadas con los principios de la mecánica clásica, con el objetivo de lograr un impacto efectivo en un sitio predeterminado.

1.2.2 Objetivos Específicos

- Refinar la estructura del robot enfocándose en la estética y la estabilidad alineándose con las funciones detalladas en el objetivo general del proyecto.
- Maximizar la eficiencia del trabajo en grupo mediante el uso de una carta Gantt.
- Implementar una interfaz con botones con la capacidad de producir movimientos y realizar el golpe por un usuario.
- Programar funciones asociadas con la interfaz.

1.3 Restricciones

- Trabajar en base a las piezas del Kit Lego Mindstorms EV3.
- Utilizar el lenguaje de programación Python y la biblioteca EV3 dev.
- Trabajo con el Kit Lego Mindstorms EV3 solo en instalaciones especificadas por el profesor.
- Cumplir con las fechas estipuladas de entregables.
- Utilizar el sistema operativo Linux.
- Toda la documentación tendrá que ser subida a la plataforma Redmine.

1.4 Entregables.

1.4.1. Archivos Redmine

- Presentaciones del proyecto.
- Bitácoras semanales.
- Informes del proyecto.
- Manual de usuario.
- Carta Gantt.

1.4.2. Robot y Código

- Interfaz final del control del robot.
- Versión final del robot.

2. Organización del Personal

Como parte del fortalecimiento de las bases de este proyecto, se asignaron roles específicos a cada miembro del equipo, buscando optimizar la eficiencia en el trabajo colaborativo. Destacando la importancia de cada rol, se delinearon responsabilidades y funciones para cada integrante, enfocándose en sus habilidades y contribuciones particulares. Esta asignación estratégica de roles se planteó con el propósito de maximizar la coordinación y el rendimiento del equipo, asegurando un flujo de trabajo efectivo y una distribución equitativa de tareas clave.

2.1 Descripción y Asignación de los Roles

Rol	**Descripción**	**Integrante(s)**
Jefe del grupo	Se encarga de organizar y supervisar al grupo.	Fabian Quezada
Ensamblador(es)	Se encarga de construir el robot de modo que funcione óptimamente y cumpla con los objetivos propuestos.	Javier Huanca Maykol Bravo
Programador(es)	Se encarga de codificar, diseñar y actualizar el código con el que se programará al robot.	Diego Ferrada Maykol Bravo
Documentador(es)	Registra y redacta los entregables (informes, bitácoras , wiki ,etc).	Fabian Quezada Joshua Jara

2.2 Mecanismos de Comunicación

Para lograr una gestión organizativa efectiva, resulta fundamental establecer un sistema de comunicación entre los miembros. A través del medio seleccionado, se pueden proponer nuevas ideas que posiblemente no hayan surgido en reuniones físicas, concretar tareas pendientes, definir nuevas responsabilidades y explorar diversas posibilidades. En el proceso de desarrollo del proyecto, se considerará utilizar la plataforma de redes sociales WhatsApp como la herramienta principal de comunicación. Esta plataforma es la más ampliamente empleada por los integrantes y cuenta con la familiaridad de gran parte del equipo en su utilización.

3. Planificación del Proyecto

3.1 Actividades

3.1.1 Documentación del proyecto

Actividad	Descripción	Miembros asignados
Carta gantt	Programa tareas, asignando tiempos y responsabilidades para coordinar eficientemente el progreso del proyecto en la plataforma redmine.	Diego Ferrada Joshua Jara
Bitacoras	Registra el progreso semanal destacando logros, obstáculos y propuestas para enriquecer el proyecto.	Javier Huanca Joshua Jara
Wiki	Presentar el proyecto mediante imágenes e información, en la plataforma redmine para una comprensión concisa y accesible del trabajo realizado.	Maykol Bravo Javier Huanca

3.1.2 Fase I

Actividad	Descripción	Miembros asignados
Verificación del material entregado	Analizar en profundidad el material con la finalidad de encontrar desperfectos o carencias.	Todo el personal
Distribución del trabajo	Designación de roles a cada integrante a lo largo del Proyecto.	Todo el personal
Lluvia de ideas	Planteamiento de la estructura del robot cumpliendo los requerimientos.	Todo el personal
Construcción del robot	Creación del robot mediante la utilización de las piezas del kit, cumpliendo las ideas previstas en la lluvia de ideas.	Maykol Bravo Fabian Quezada
Creación del informe I	Redactar un informe desarrollando los siguientes puntos: Panorama General.	Javier Huanca Diego Ferrada

	Organización del personal. Planificación del Proyecto. Planificación de recursos.	
Creación de presentación I	Confeccionar un PPT utilizando el contenido del informe y los requerimientos de las pautas.	Fabian Quezada

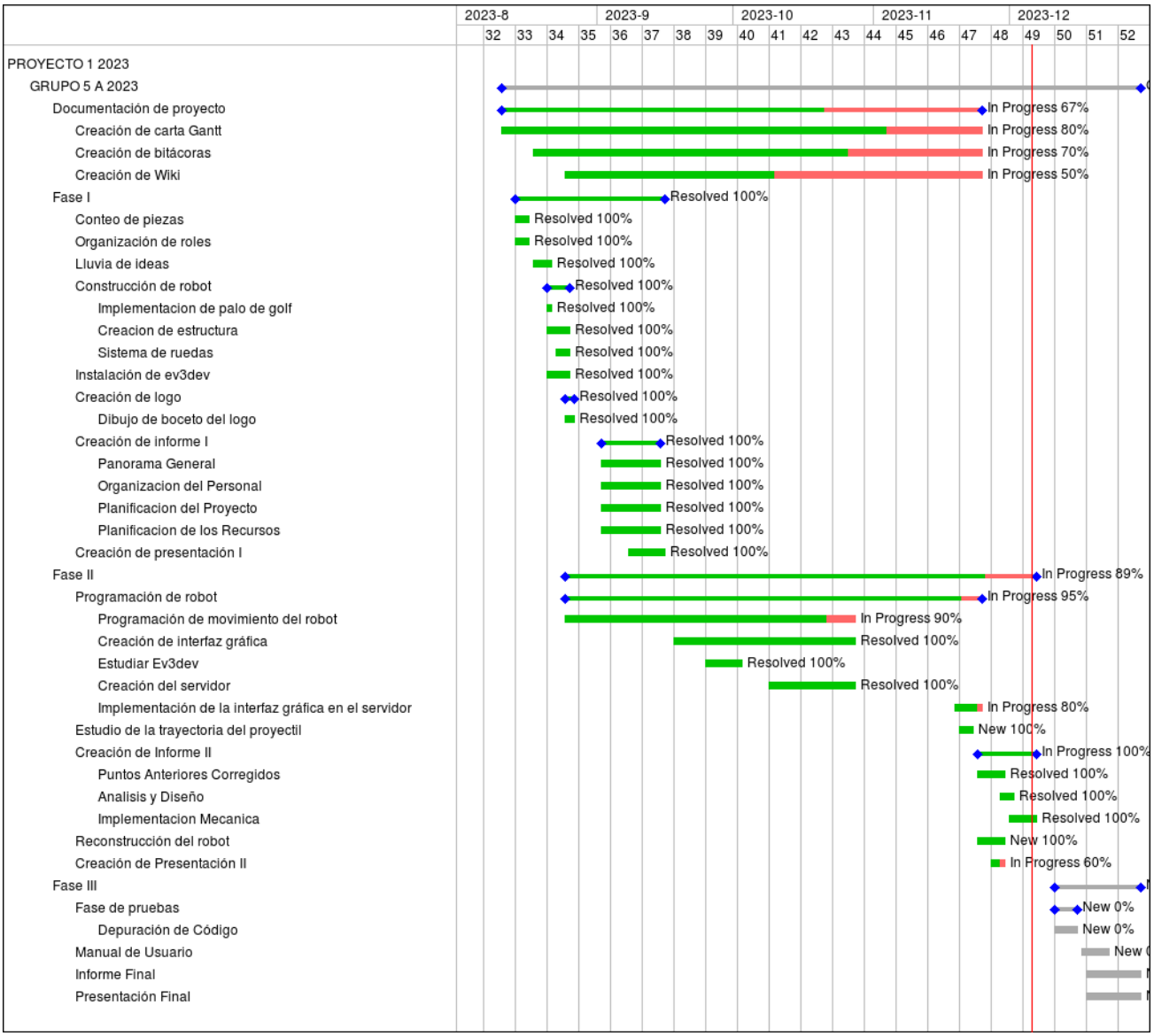
3.1.3 Fase II

Actividad	Descripción	Miembros asignados
Programación de interfaz y del servidor.	Elaborar el código con funciones requeridas para cumplir los criterios, priorizando precisión y funcionalidad, ajustándose a los requisitos del proyecto de manera integral.	Diego Ferrada Maykol Bravo Javier Huanca
Creación del informe II	Redactar un informe desarrollando los siguientes puntos: Corrección de errores. Análisis y diseño. Implementación Mecánica.	Fabian Quezada Joshua Jara
Estudio de la trayectoria del proyectil	Realizar un análisis exhaustivo del tiro parabólico de una pelota, calculando su trayectoria y analizando su movimiento en relación con la gravedad y otros factores físicos relevantes.	Joshua Jara Fabian Quezada
Reestructuración del robot	Revisar y corregir fallos previos en la reconstrucción del robot, priorizando mejoras estructurales y ajustes funcionales para optimizar su rendimiento y precisión.	Maykol Bravo Diego Ferrada
Creación de presentacion II	Elaborar un PPT utilizando el contenido del informe y los requerimientos de las pautas.	Joshua Jara Fabian Quezada

3.1.4 Fase III

Actividad	Descripción	Rol Asignado
Fase de pruebas	Testeo de las funcionalidades del robot tales como su movimiento, fuerza de golpe y la eficacia de la conexión remota.	Diego Ferrada
Manual de usuario	Desarrollar una guía sencilla que describa el funcionamiento del robot en conjunto con la interfaz de usuario.	Fabian Quezada
Informe final	Finalizar la redacción.	Joshua Jara Fabian Quezada
Presentacion final	Elaborar un PPT utilizando el contenido del informe y los requerimientos de las pautas.	Joshua Jara Fabian Quezada

3.2 Asignación de Tiempo



3.3 Gestión de Riesgos

1. Catastrófico
2. Crítico
3. Circunstancial
4. Irrelevante

Riesgos	Probabilidad de concurrencia	Nivel de impacto	Acción remedial
La falta de piezas para el armado del robot.	20%	2	Encontrar una forma para trabajar con las piezas disponibles o ir a buscar a la sala de ayudantía.
Batería agotada del EV3.	60%	3	Cargar completamente la batería antes de cada uso.
Fallo mecánico sobre los motores y ruedas.	5%	2	Realizar un mantenimiento regular y reemplazar las piezas desgastadas.
Fallo en la conectividad entre el robot y el computador.	30%	1	Asegurarnos que los controladores y software estén instalados y funcionen correctamente.
Pérdida de la codificación del robot.	5%	1	Almacenar una copia de los programas en caso de una posible pérdida.

4. Planificación de los Recursos

4.1 Hardware

- Kit LEGO Mindstorms EV3.
- Notebook.
- Tarjeta microSD.
- Conexión SSH.
- Piezas extras.

4.2 Software

- Lenguaje Python 3.
- Software Ev3dev.
- IDE Visual Studio Code.
- Plataforma Redmine.
- Microsoft Office

4.3 Estimación de Costos

4.3.1 Costos de Hardware

Descripción	**Cantidad**	**Costo (CLP)**
Kit LEGO Mindstorms EV3	1	$1.230.000
NoteBook Dell	1	$400.000
NoteBook Lenovo V14	2	$990.000
Tarjeta MicroSD	1	$10.000
Piezas extras	ind.	$8.500
Costo Total		$3.628.500

4.3.2 Costos de Software

Descripción	Cantidad	Costo (CLP)
Licencia Office 365 (Mensual)	1	$6.990/mes
Discord	-	Gratuito
Visual Studio Code	-	Gratuito
Whatsapp	-	Gratuito
Costo Total		$34.950

4.3.3 Costos de Personal

Rol	Cantidad	Valor Hora	Horas Totales	Total (5 meses)
Jefe de Grupo	1	$6.000	180	$1.080.000
Ensamblador	2	$4.000	180	$720.000(c/u)
Documentador	2	$4.000	180	$720.000(c/u)
Programador	2	$5.000	180	$900.000(c/u)
Costo Total				$5.760.000

4.3.4 Costo total del Proyecto

Descripción	Total
Software	$34.950
Hardware	$3.628.500
Personal	$5.760.000
Costo Total	**$9.423.450**

5. Análisis y Diseño

5.1 Especificación de Requerimientos

5.1.1 Requerimientos Funcionales

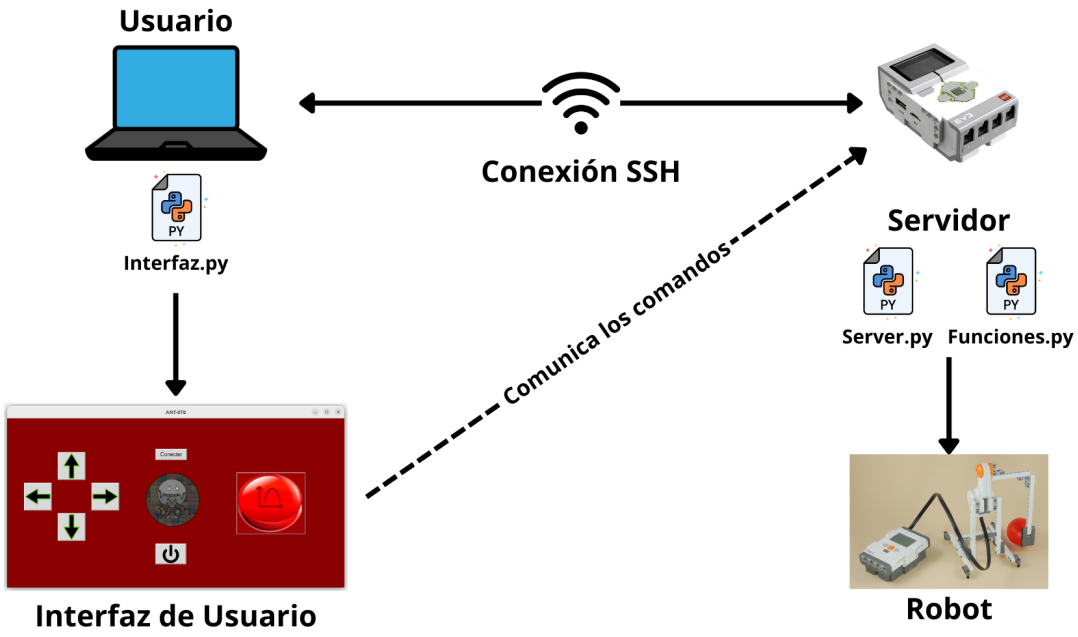
- El robot debe tener la capacidad de moverse.
- El robot debe tener la capacidad de golpear una pelota de golf en una trayectoria específica.
- El robot debe ser controlado remotamente mediante una interfaz de usuario.

5.1.2 Requerimientos no Funcionales

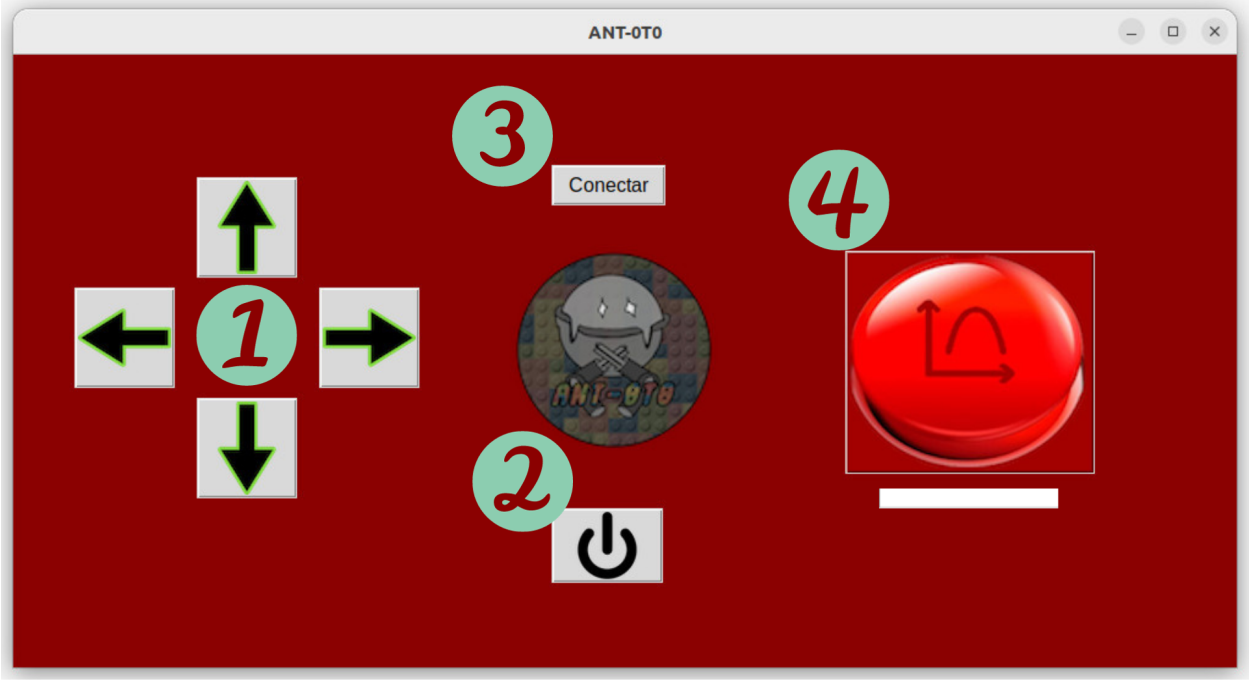
- Interfaz gráfica que contenga los controles necesarios para poner al robot en movimiento, conectarse al dispositivo controlador y golpear la pelota.
- El programa debe ser desarrollado con el lenguaje de programación Python en conjunto con la biblioteca ev3dev.
- El robot debe ser ensamblado utilizando piezas del kit Lego Mindstorms.

5.2 Arquitectura

El proceso de conexión y control remoto del kit Lego Mindstorms EV3 se realiza desde un sistema operativo Linux, la comunicación se establece a través de una conexión SSH. Tras configurar la red, la conexión se inicia mediante la terminal Linux con el comando ssh robot@ev3dev.local. Una vez autenticado, se accede a la terminal del EV3, permitiendo la ejecución de comandos y programas en Python. Luego de conseguir e implementar la dirección ip del robot se ejecuta en éste el servidor que recibe la información del usuario.
Además, se implementa una interfaz gráfica en un archivo Python en el PC Linux, desde donde se envían instrucciones remotas al robot a través de la conexión SSH.
En el siguiente diagrama se ejemplifica el proceso de conexión:



5.3 Interfaz



1. Botones de movimiento, cada flecha es un botón que pone en movimiento al robot en la dirección correspondiente.
2. Botón de apagado, al ser presionado cierra el programa.
3. Botón conectar, intenta establecer una conexión entre el servidor, el robot y el usuario.
4. Botón lanzar, al presionarlo utiliza la distancia proporcionada en la caja de texto inferior, de lo contrario, lanza la pelota a una distancia aproximada de un metro.

6. Implementación

6.1 Fundamentos de Proyectiles

El tipo de movimiento que sigue la pelota de golf al ser lanzada es un movimiento parabólico ideal, trayectoria que se ha estudiado previamente en la asignatura de Mecánica Clásica. Este tipo de movimiento se estudia exclusivamente en dos dimensiones, caracterizándose por la ausencia de fuerzas oponentes al movimiento a lo largo del eje x, lo que mantiene la velocidad constante en esa dirección. En contraste, en el eje y, la variación del movimiento experimenta cambios uniformes, siendo influenciada por la constante aceleración de la gravedad en cada instante.

La configuración inicial implica un ángulo de inclinación específico, en este caso se asume un ángulo de aproximadamente 45°. Además, se toma como dato conocido la distancia entre el objetivo y la posición de la pelota, junto con la altura inicial de la pelota.
Ejemplo de la trayectoria:

- *Ángulo inicial: 45°.*
- *Distancia entre la pelota y el objetivo: 1m.*
- *Altura inicial de la pelota: 0,08 m.*

Sea θ *el ángulo de tiro,* v_i *la velocidad inicial de la pelota e* y_0 *la altura inicial de la pelota:*

El desglose de la velocidad en cada eje es:

La velocidad en el eje x: $v_x = v_i \cos\theta$

La velocidad en el eje y: $v_y = v_i \operatorname{sen}\theta$

Por lo tanto, las ecuaciones de posición son:

La posición en el eje x: $x = v_i \cos\theta \cdot t$

La posición en el eje y: $y = y_0 + v_i \operatorname{sen}\theta \cdot t - \frac{1}{2} g t^2$ *Donde:* $y_0 = 0,08\ m$

- *Condiciones para el encuentro:* $x = 1\ ;\ y = 0$

Reemplazando:

$$1 = v_i \cos\theta \cdot t$$

$$0 = 0,08 + v_i \operatorname{sen}\theta \cdot t - 4,9 t^2$$

Despejando v_i ***:***

$$v_i = \frac{1}{\cos\theta \cdot t}$$

Luego, reemplazamos en la ecuación de posición en y:

$$0 = 0,08 + \frac{sen\,\theta}{cos\,\theta} - 4,9t^2$$

Despejando t:

$$t^2 = \frac{0,08 + tan\,\theta}{4,9}$$

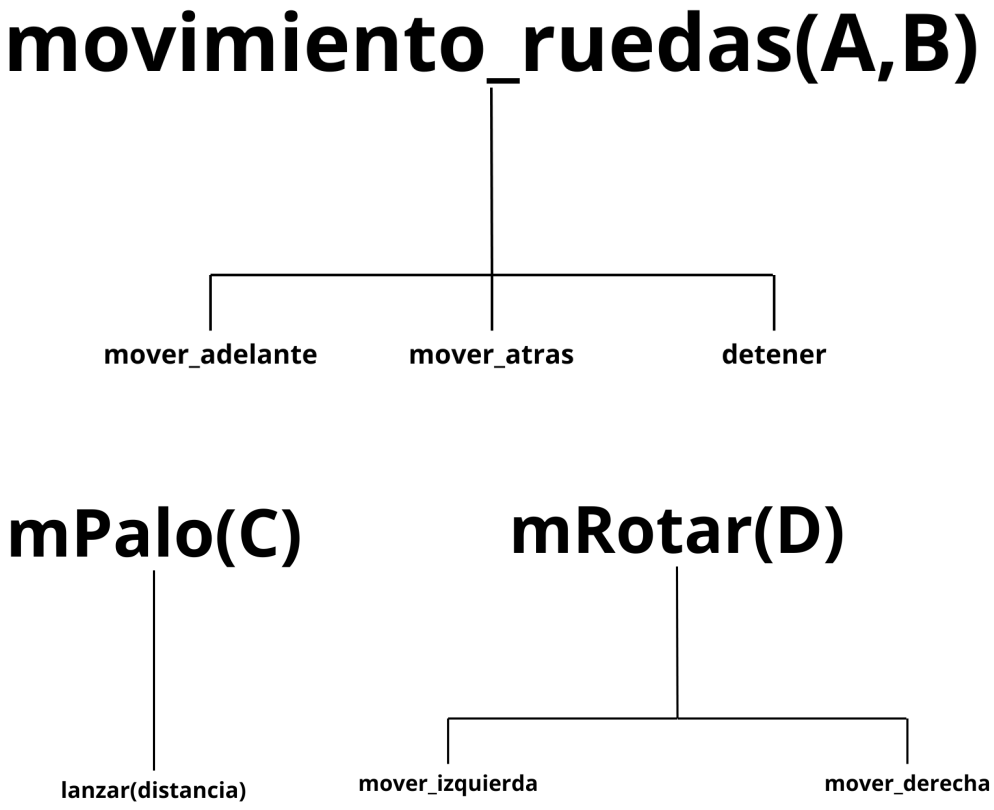
$$t = \sqrt{\frac{0,08 + tan\,\theta}{4,9}} \approx 0,58\ s$$

Finalmente reemplazamos en la primera ecuación:

$$v_i = \frac{1}{cos\,\theta \cdot \sqrt{\frac{0,08 + tan\,\theta}{4,9}}} \approx 3,23\ m/s$$

Por lo tanto, cuando el objetivo se encuentra a 1 m la velocidad inicial necesaria es $v_i \approx 3,23$ *y el tiempo de vuelo es* $t \approx 0,58\ s$.

6.2 Diagramas



6.3 Descripción de los Programas

- ### 6.3.1 Funciones

  En el archivo “funciones.py” se hallan establecidas las funciones responsables de interactuar con el robot. Asimismo, en él se encuentran especificadas cada una de las funciones que el robot es capaz de realizar.

```
from ev3dev2.motor import LargeMotor, MediumMotor, OUTPUT_A, OUTPUT_B, OUTPUT_C, OUTPUT_D, SpeedRPM, MoveTank
from ev3dev2.sound import Sound
import math
import os
import sys
from time import sleep

#movimiento simultáneo de las ruedas(rueda izquierda, rueda derecha)
movimiento_ruedas = MoveTank(OUTPUT_A, OUTPUT_B)
#movimiento del palo del robot
mPalo = LargeMotor(OUTPUT_C)
#movimiento de rotación de las ruedas delanteras
mRotar = MediumMotor(OUTPUT_D)


#función para lanzar un objeto con el palo
def lanzar(distancia):
    pi = math.pi
    vel_necesaria = math.sqrt((4.9*distancia)/math.cos(pi/4)*math.sin(pi/4))
    vel_rpm = (60*vel_necesaria)/(2*pi*0.189)
    # Rotar el motor en -180 grados al 6% de velocidad
    mPalo.on_for_degrees(6,-180)
    # Rotar el motor en 225 grados al porcentaje de velocidad de vel_rpm
    mPalo.on_for_degrees(SpeedRPM(round(vel_rpm)),225)
    # Rotar el motor en -45 grados al 6% de la velocidad
    mPalo.on_for_degrees(6,-45)

#función para mover el robot hacia adelante
def mover_adelante():
    movimiento_ruedas.on(-100, -100)

#función para mover el robot hacia atras
def mover_atras():
    movimiento_ruedas.on(100, 100)

#función para mover el robot hacia la izquierda
def mover_izquierda():
    mRotar.on(50)

#función para mover el robot hacia la derecha
def mover_derecha():
    mRotar.on(-50)

#función para detener el robot
def detener():
    movimiento_ruedas.stop()
```

- ### 6.3.2 Servidor

En el archivo “servidor.py" se encarga de posibilitar la conexión a través de un servidor creado por la librería socket de Python, su tarea principal consiste en leer las instrucciones que son enviadas desde un cliente. Estas instrucciones están organizadas por caracteres dentro de un bucle que se repite constantemente hasta que la conexión finaliza.

```
import socket
from funciones import *

#se crea un socket
s = socket.socket()
s.settimeout(5)
print("Socket creado")
port = 2222
#se asocia una dirección IP y un número de puerto al socket
s.bind(("", port))
print("El socket se creo con puerto:{}".format(port))
#se comienza a escuchar conexiones entrantes (máximo 5 conexiones pendientes)
s.listen(5)
print("The socket is listening....")
#se acepta la conexión, creando un nuevo socket a partir de la conexion entrante y la dirección del cliente
connect, addr = s.accept()
print("Se conecto a {}".format(addr))
while True:
    #se define el carácter que el socket del cliente recibe como dato
    rawByte = connect.recv(1)
    #se traduce el carácter recibido
    char = rawByte.decode('utf-8')
    if (char == 'w'):
        mover_adelante()
    if (char == 's'):
        mover_atras()
    if (char == 'a'):
        mover_izquierda()
    if (char == 'd'):
        mover_derecha()
    if (char == ' '):
        detener()
    if (char == '1'):
        lanzar(2)
```

- ### 6.3.3 Interfaz

En el archivo “interfaz.py” se implementa una interfaz gráfica usando la biblioteca de Tkinter de Python para controlar el robot. Incluye controles direccionales(adelante, atrás, izquierda, derecha), un botón de apagado, un botón para lanzar una acción y un botón para establecer la conexión con el robot.

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from tkinter import *
import socket
import sys

ev3_ip = "192.168.70.147"

#Ventana
root = tk.Tk()
root.config(width=1000, height=500)
root.config(bg="darkred")
root.title("ANT-0T0")

#Acciones del robot
def irAdelante():
    clientSocket.send(bytes([ord('w')]))

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

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

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

def inicioLanzar():
    clientSocket.send(bytes([ord('l')]))

def soltar(event):
    clientSocket.send(bytes([ord(' ')]))

def conectar(address):
    try:
        clientSocket.connect((address,port))
        messagebox.showinfo("Mensaje Servido","Cliente conectado al robot: {0} : {1}".format(address,port))
    except socket.error:
        messagebox.showwarning("Conexión erronea","No se ha logrado al conexion, verifique la ip {0}".format(address))
        clientSocket.close()
```

```
#Imagenes
imgLogo = tk.PhotoImage(file="logo_ant0t0.png")
imgFlechaIzq = tk.PhotoImage(file="flecha_izquierda.png")
imgFlechaSup = tk.PhotoImage(file="flecha_superior.png")
imgFlechaDer = tk.PhotoImage(file="flecha_derecha.png")
imgFlechaInf = tk.PhotoImage(file="flecha_inferior.png")
imgApagar = tk.PhotoImage(file="Boton_Apagar.png")
imgLanzar = tk.PhotoImage(file="imgLanzar.png")

etiqueta_logo = tk.Label(root, image=imgLogo, bg = "darkred").place(x=350,y=150)
#Boton Flecha Izquierda
Bt_FlechaIzq = Button(root, image=imgFlechaIzq, command=irIzquierda)
Bt_FlechaIzq.place(x=50,y=190)
#Boton Flecha Superior
Bt_FlechaSup = Button(root, repeatdelay=50, repeatinterval=50, image=imgFlechaSup, command=irAdelante)
Bt_FlechaSup.place(x=150,y=100)
Bt_FlechaSup.bind('<ButtonRelease-1>', soltar)
#Boton Flecha Derecha
Bt_FlechaDer = Button(root, image=imgFlechaDer, command=irDerecha)
Bt_FlechaDer.place(x=250,y=190)
#Boton Flecha Inferior
Bt_FlechaInf = Button(root, repeatdelay=50, repeatinterval=50, image=imgFlechaInf, command=irAtras)
Bt_FlechaInf.place(x=150,y=280)
Bt_FlechaInf.bind('<ButtonRelease-1>', soltar)

#Boton Apagado
Bt_Apagar = Button(root, image=imgApagar, command=root.destroy).place(x=440,y=370)

#Boton Lanzar
Bt_Lanzar = Button(root, image=imgLanzar, bg = "darkred", command=inicioLanzar)
Bt_Lanzar.place(x=670, y=160)

#Boton Conectar
button_connect = tk.Button(root, text="Conectar", command= lambda: {conectar(ev3_ip)}, font=("Arial",12)).place(x=440,y=90)

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(ev3_ip))

clientSocket = socket.socket()
port = 2222

root.mainloop()
```

7. Resultados

7.1 Estado Actual del Proyecto

- Versión del robot “ANT-0T0” semi finalizada (Estructura final susceptible a pequeñas modificaciones).
- Sistema de movimiento terminado, giro en direcciones laterales, aceleración y reversa finalizados.
- Sistema de disparo casi finalizado, palo de golf terminado, base para la pelota a modificar.
- Informe corregido en su totalidad según la retroalimentación del informe de planificación.
- Conexión al servidor del robot finalizada.
- Código del robot finalizado, funciones de movimiento y giro listas para la fase de pruebas.
- Interfaz gráfica finalizada, realizada con la librería tkinter en Python.
- Actualización de archivos y wiki en redmine.
- Mejora a la Carta Gantt.

7.2 Problemas Encontrados y Solución Propuesta

Problema Encontrado	**Solución Propuesta**
Problemas de estabilidad en la base del robot	Quitar piezas innecesarias del robot, re-distribuir el peso y añadir estructuras que mejoren la estabilidad
Problemas con el sistema de giro del robot	Reemplazar el sistema de giro que utilizaba dos motores para redireccionar las ruedas por uno de un motor que utiliza engranajes para realizar el giro
Base inadecuada para la pelota de golf	Reconstruir la base para la pelota por una que no interfiera con el paso del palo que golpea la pelota
Problemas de conectividad con el servidor	Realizar apropiadamente la conexión mediante ssh y ejecutar el archivo que contiene el servidor directamente desde el robot
Adaptabilidad a los nuevos roles	Realizar una reunión de retroalimentación

	sobre el progreso de cada integrante en su anterior rol y establecer las tareas futuras en los nuevos roles

8. Conclusión

En conclusión, el proyecto "ANT-0T0" ha sido un logro integral que destaca la aplicación efectiva de conceptos teóricos en la construcción de un robot de minigolf utilizando el kit LEGO Mindstorms EV3. La utilización de la conexión SSH facilitó la conexión remota con el robot, mientras que el desarrollo de la interfaz gráfica permitió un control intuitivo. La planificación meticulosa, expresada en la carta Gantt y la plataforma Redmine, demostró ser esencial para una gestión efectiva del tiempo y los recursos.

El proceso de construcción involucró la creación de diversos prototipos, cada uno refinando la estructura y funcionalidad del robot. Se llevaron a cabo ajustes significativos, incluida la reconstrucción del sistema de giro del robot, para optimizar su estabilidad. Este enfoque iterativo condujo al desarrollo del prototipo final, listo para las pruebas finales.

En resumen, "ANT-0T0" ha representado un desafío integral que ha permitido integrar teoría y práctica, fortalecer habilidades técnicas y de gestión de proyectos, y alcanzar con éxito los objetivos propuestos.

9. Referencias

- *Python Programming Language*. Python.org. (n.d.). https://www.python.org
- ev3dev (2020). ev3dev [Online]. Available: https://www.ev3dev.org/
- Visual Studio Code https://code.visualstudio.com/
- Linux Ubuntu https://ubuntu.com/
- LEGO MINDSTORMS EV3 https://www.lego.com/en-us/product/lego-mindstorms-ev3-31313