UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

TIGER BOT
COMPANY

PROYECTO FINAL 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.

Wiki

Índice

Introducción y Planificación

Tiger Bot

Índice

Análisis y Diseño

Requerimientos

Funcionales	No Funcionales
El robot debe poder moverse	La interfaz gráfica debe contener los movimientos del (avanzar, retroceder, girar izquierda, girar derecha) ac
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 M
El golpe del robot debe generar movimiento parabólico en la pelota	El robot debe poder ser controlado desde un computa
	La interfaz gráfica debe ser de fácil entendimiento par

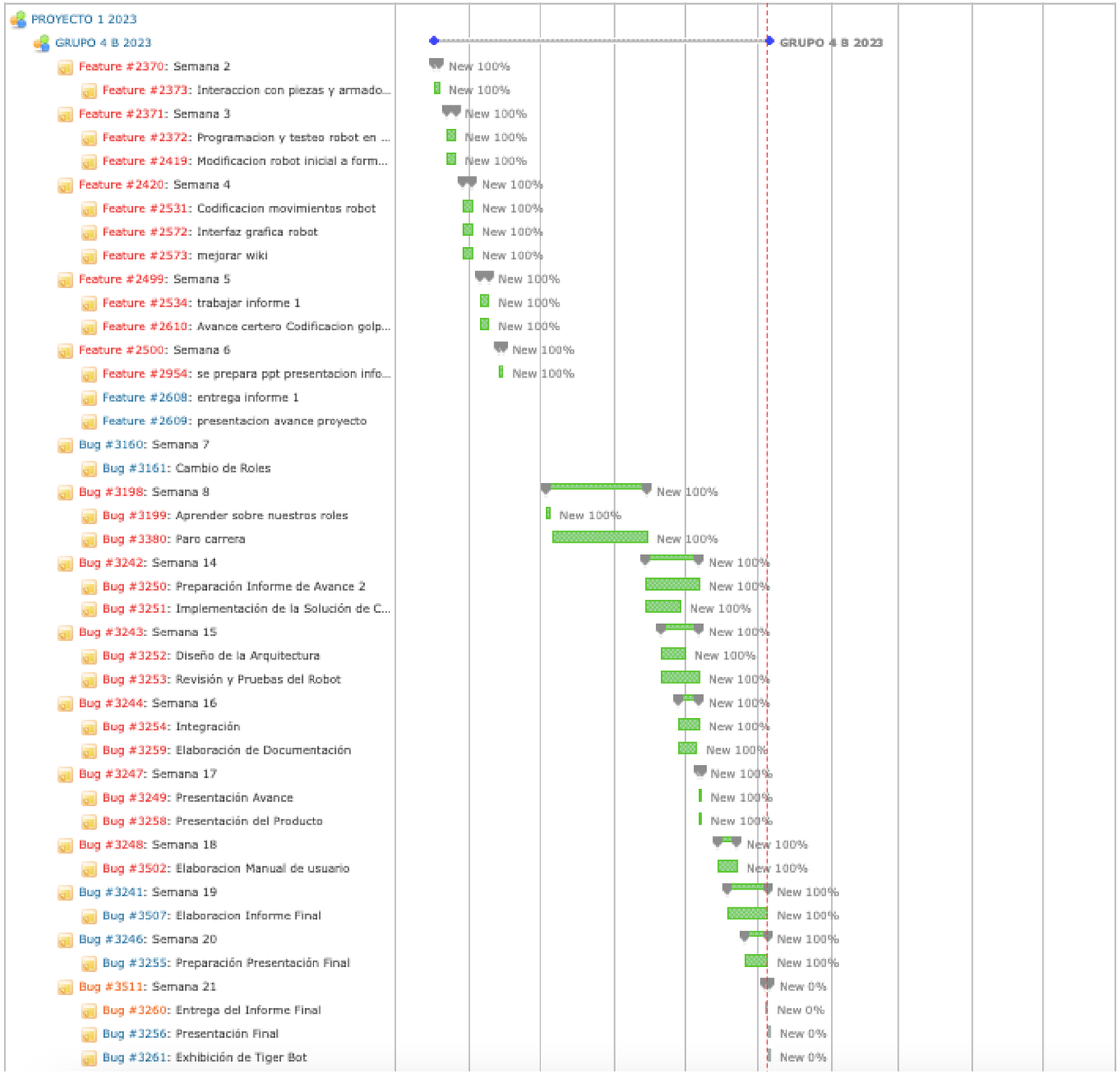
Arquitectura

Resultados

Actualmente nuestro proyecto cuenta con:

- La wiki del proyecto
- Informe avance formulado
- La conexión vía remota robot
- Carta Gantt actualizada
- Bitácoras subidas (durante paro trabajamos en robot y seguimos subiendo bitácoras)
- La interfaz gráfica de movimientos y control del robot la cual fue programada con la librería "tkinter"
- Servidor de conexión del robot con interfaz gráfica implementada con librería de python "socket"
- Funciones de movimiento y desplazamiento del robot además del golpe de la pelota de golf con el bra
- Estructura base apoyo pelota golpe de golf
- Control movimientos robot a través de mando tanto xbox como playstation

Asignación de Tiempo



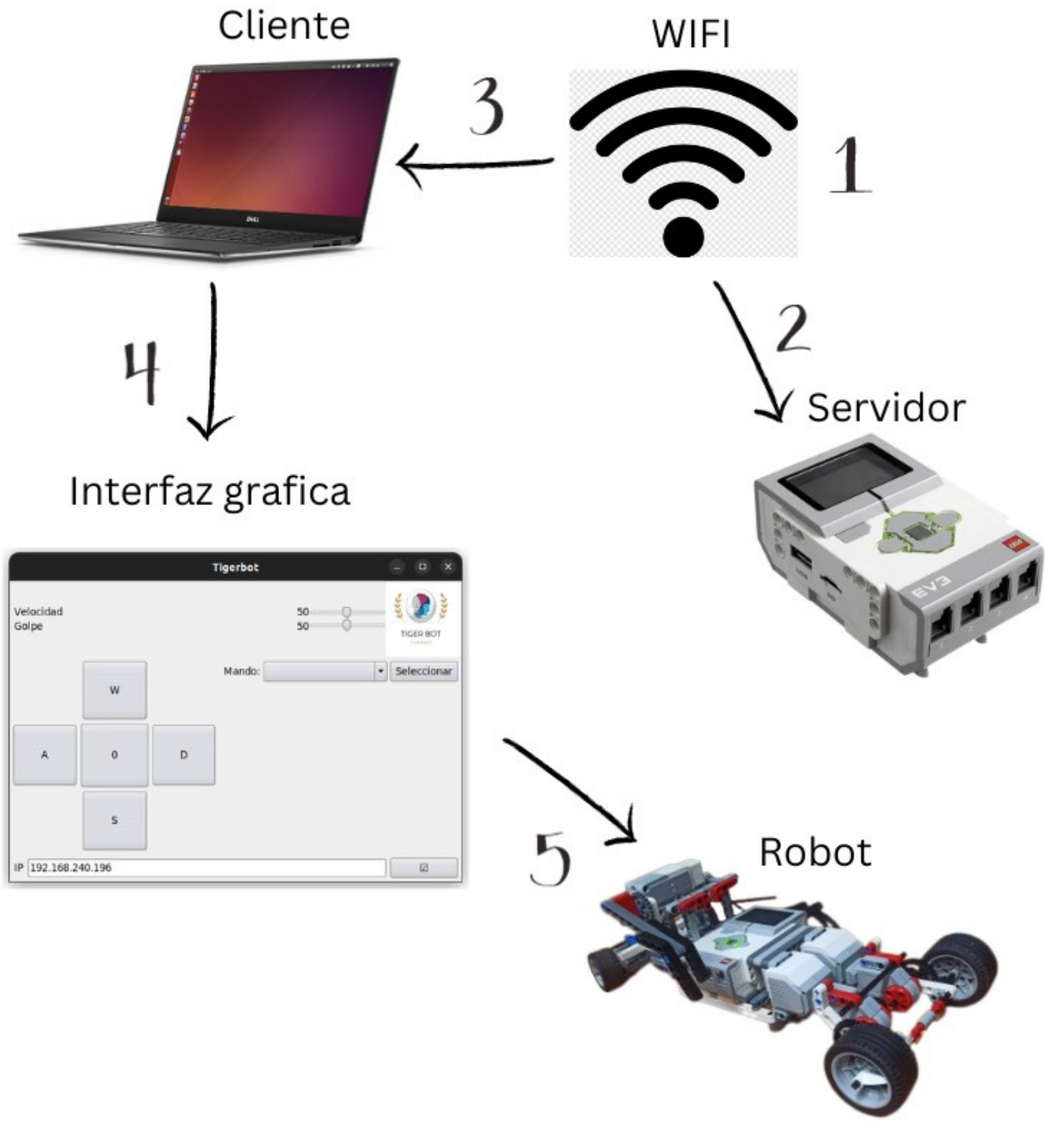
Estimación de Costos

Costos Totales

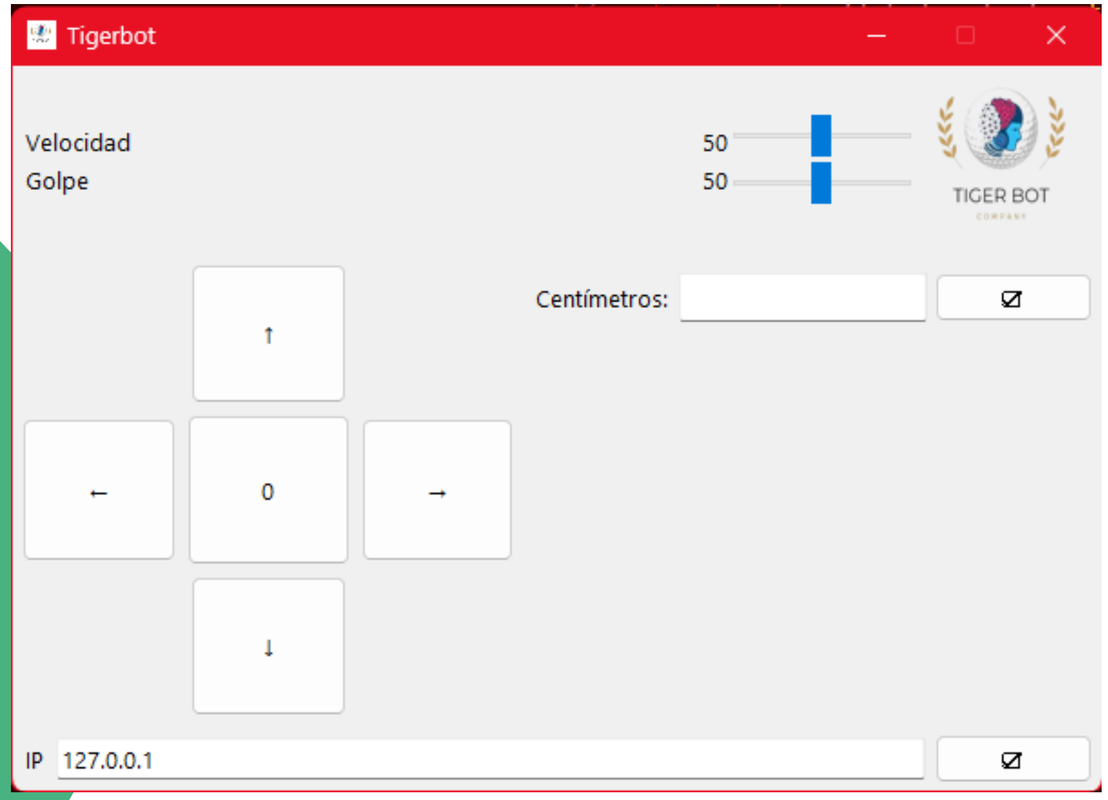
Total	Costo Total
Costo Hardware	$1,640,656
Costo Software	$0
Costo Empleados	$5,901,750
Total (21 semanas)	**$7,542,406**

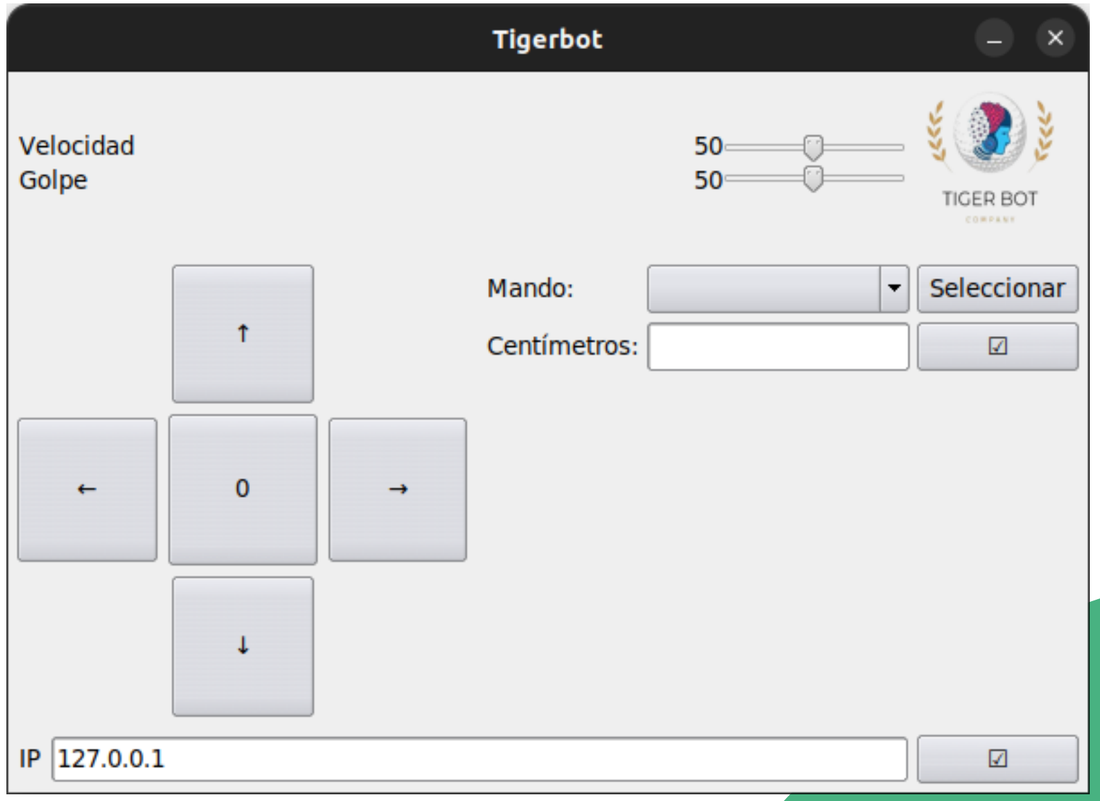
Análisis y Diseño

Arquitectura



Interfaz


Tigerbot
Velocidad
Golpe
50
50
TIGER BOT
Centímetros:
↑
←
0
→
↓
IP
127.0.0.1


Tigerbot
Velocidad
Golpe
50
50
TIGER BOT
Mando:
Seleccionar
Centímetros:
↑
←
0
→
↓
IP
127.0.0.1

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
for name, widget in self.control.items():
    widget:ttk.Button
    if name == "up":
        button_command(widget,
        lambda _: (self.robot.avanzar(self.velocidad['movimiento']['value'].get()) if self.robot.is_alive() else ...),
        lambda _: (self.robot.avanzar(0) if self.robot.is_alive() else ...))
    elif name == "left":
        button_command(widget,
        lambda _: (self.robot.izquierda(int(self.velocidad['movimiento']['value'].get()/3)) if self.robot.is_alive() else ...),
        lambda _: (self.robot.izquierda(0) if self.robot.is_alive() else ...))
    elif name == "down":
        button_command(widget,
        lambda _: (self.robot.retroceder(self.velocidad['movimiento']['value'].get()) if self.robot.is_alive() else ...),
        lambda _: (self.robot.retroceder(0) if self.robot.is_alive() else ...))
    elif name == "right":
        button_command(widget,
        lambda _: (self.robot.derecha(int(self.velocidad['movimiento']['value'].get()/3)) if self.robot.is_alive() else ...),
        lambda _: (self.robot.derecha(0) if self.robot.is_alive() else ...))
    elif name == "center":
        button_command(widget,
        lambda _: (self.robot.atacar(self.velocidad['golpe']['r_value'].get()) if self.robot.is_alive() else ...),
        lambda _: (self.robot.perdonar() if self.robot.is_alive() else ...))
```

```python
def mecánica_click():
    try:
        if self.robot.is_alive():
            valor = float(self.mecánica['entry'].get())/100
            valor = dis2vel(valor)
            self.robot.atacar(valor)
            self.velocidad['golpe']['w_value'].set(valor)
        else:
            print("Robot no Conectado")
    except ValueError:
        print("Entrada inválida")
    except Exception as e:
        print(e)
```

```python
def vel2dis(x) -> float:
    if not (50 <= x <= 100):
        raise Exception(f"Valor fuera de rango")
    return -9*(x-1825/18)**2/62500+2881/3600
```

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...")
```

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()

```

Fundamentos a Proyectiles

Calculos

Distancia Recorrida

$x_F = x_i + v_x \cdot t$ // $- x_i$

$\Delta x = v_x \cdot t = d$ // Reemplazando

$$d = \frac{v_x\left(-v_y - \sqrt{2g\Delta y + v_y^2}\right)}{g} \quad //$$

Ejemplo:

$\theta = 45°$, $y_F = 0$, $y_i = 0{,}05\,m$, $g = -9{,}8\,m/s^2$

$v_i = 20\,m/s$

$$d[m] = \frac{-20\cos(45°)\left[20\,\mathrm{sen}(45°) + \sqrt{-9{,}8 \cdot 2 \cdot (-0{,}05) + (20\,\mathrm{sen}\,45)^2}\right]}{-9{,}8}$$

$d = 40{,}86\,m$

Calculo Velocidad Inicial

$$d = \frac{-v_x\left(v_y + \sqrt{2g\Delta y + v_y^2}\right)}{g} \quad // \cdot g$$

$$dg = -v_x v_y - v_x\sqrt{2g\Delta y + v_y^2} \quad // + v_x v_y$$

$$dg + v_x v_y = -v_x\sqrt{2g\Delta y + v_y^2} \quad //()^2$$

$$(dg)^2 + 2dg\,v_x v_y + (v_x v_y)^2 = v_x^2(2g\Delta y + v_y^2)$$

$$(dg)^2 + 2dg\,v_x v_y + (v_x v_y)^2 = v_x^2 v_y^2 + v_x^2 \cdot 2g\Delta y$$

$$// - v_x^2 v_y^2$$

$$(dg)^2 + 2dg\,v_i\cos(\theta)\,v_i\,\mathrm{sen}\,\theta = v_i^2\cos^2\theta \cdot 2g\Delta y$$

$$// - v_i^2\cos^2\theta \cdot 2g\Delta y - (dg)^2$$

$$2dg\,v_i^2\cos\theta\,\mathrm{sen}\,\theta - 2g\Delta y\,v_i^2\cos^2\theta = -(dg)^2$$

$$2v_i^2 g\cos\theta\left(d\,\mathrm{sen}\,\theta - \Delta y\cos\theta\right) = -(dg)^2$$

$$v_i^2 = \frac{-d^2 g^2}{2g\cos\theta(d\,\mathrm{sen}\,\theta - \Delta y\cos\theta)} \quad // \sqrt{\cdot}$$

$$v_i = \sqrt{\frac{-\Delta x^2 g}{2\cos\theta(\Delta x\,\mathrm{sen}\,\theta - \Delta y\cos\theta)}}$$

Calculos

Tiempo de vuelo, a partir de vel., alcance y Δy

$$y_F = y_i + v_y t + g t^2/2 \qquad // - y_i$$

$$\Delta y = v_y t + g t^2/2 \qquad // \cdot 2/g$$

$$\frac{2\Delta y}{g} = t^2 + \frac{2 v_y t}{g} \qquad // + \left(\frac{v_y}{g}\right)^2$$

$$t^2 + \frac{2 v_y t}{g} + \left(\frac{v_y}{g}\right)^2 = \frac{2\Delta y}{g} + \left(\frac{v_y}{g}\right)^2$$

$$\left(t + \frac{v_y}{g}\right)^2 = \frac{1}{g^2}\left(2\Delta y \cdot (g) + v_y^2\right) \qquad // \sqrt{\ }$$

$$t + \frac{v_y}{g} = \pm \frac{1}{g}\sqrt{2 g \Delta y + v_y^2} \qquad // - \frac{v_y}{g}$$

$$t = \frac{-v_y \pm \sqrt{2 g \Delta y + v_y^2}}{g}, \quad t \geq 0$$

$$t = \frac{-v_y - \sqrt{2 g \Delta y + v_y^2}}{g} \qquad // \text{(Raiz Negativo)}$$

Calculos

$$Tv = (Vy+\sqrt{(Vy^2-2*G*\Delta Y)})/G$$

Tv: Tiempo de vuelo que toma llegar al Y final en la caída

$$\Delta X = Vx*(Vy + \sqrt{(Vy^2 - 2*G*\Delta Y)})/G$$

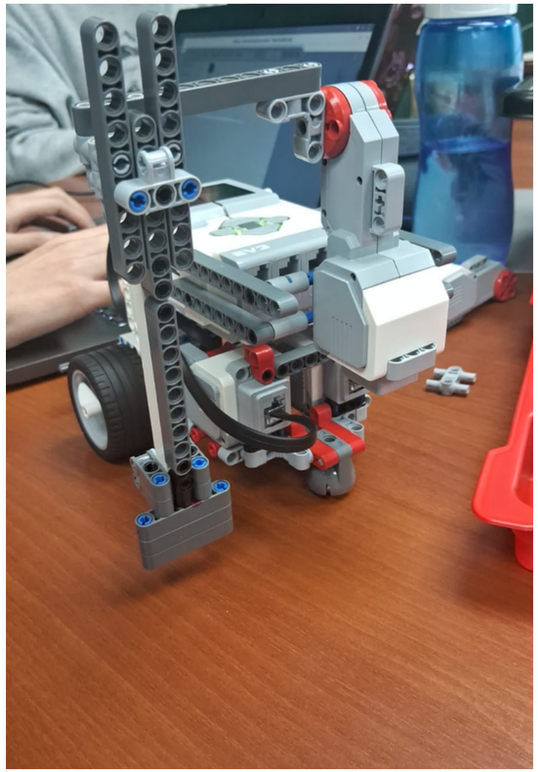
Distancia Recorrida

$$Vi = \sqrt{(G*(\Delta X)^2/(2*\cos(\theta)*(\Delta X*\sin(\theta)-\Delta Y*\cos(\theta))))}$$

Vi = Velocidad inicial

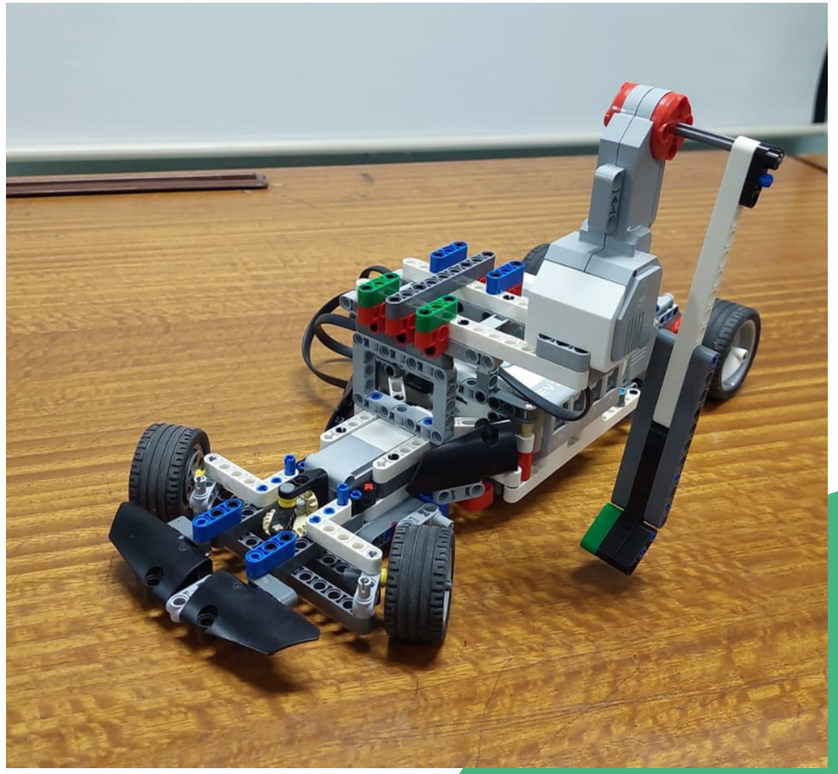
Evolución

TB1



TB2

TB3



Pruebas

Pruebas Realizadas	Pruebas Resultados
En un primer momento realizamos el golpe de golf con la pelota “base” la cual era bastante pesada	Al ser la pelota “base” bastante pesada, nos dimos cuenta de que el movimiento parabólico no se llevaba a cabo en la práctica por lo que necesitábamos una plataforma para la pelota.
El segundo intento fue ya con la plataforma hecha, la cual nos permite el movimiento parabólico pero de una manera bastante mínima en la cual con suerte se elevaba la pelota.	Al elevarse poco la pelota, nos dimos cuenta que necesitábamos una pelota más ligera que la pelota “base”, por lo que acudimos a una pelota más ligera en este caso una pelota de ping pong.
Al cambiar la pelota por una de ping pong y utilizar nuestra plataforma de lego, el movimiento parabólico fue completamente exitoso en la práctica	Ajustando las distintas potencias del brazo logramos hacer el tiro parabólico de golf solicitado, esta vez ya mucho más perfeccionado y controlado que en los intentos anteriores.

Manual de Usuario

Conclusión