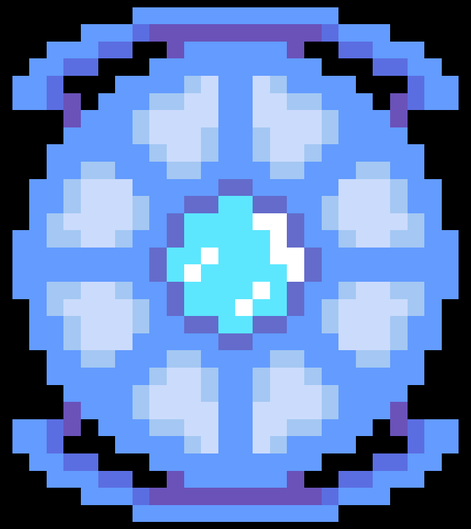


PROYECTO:
"WHEATLEY ROBOTICS"

FASE 2.

INTEGRANTES:
RENATO ALMEYDA
MARTÍN CASTILLO
OSVALDO COSTAGLIOLA
ADIEL ESPINOZA
NICOLÁS ZARZURI

TABLA DE CONTENIDOS

1. ANÁLISIS Y DISEÑO

ESPECIFICACIÓN DE REQUERIMIENTOS

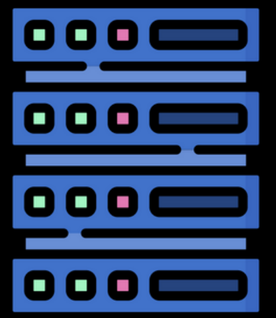
FUNCIONALES

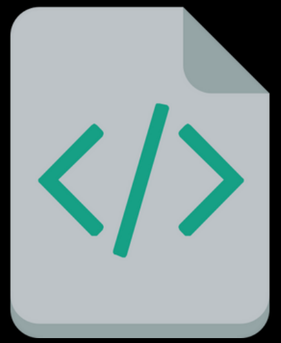
- DESARROLLAR UN ROBOT QUE SE COMUNIQUE VÍA WIFI Y SE CONTROLE MEDIANTE UNA GUI.
- CAPACIDAD PARA DESPLAZARSE Y LEVANTAR UNA BOLA DE PING PONG.
- LA GUI DEBE OFRECER OPCIONES PARA DESPLAZARSE, MOVER LA PALA DE LA BOLA Y REALIZAR EL LEVANTAMIENTO.

NO FUNCIONALES

- EL PROYECTO DEBE INCLUIR UN MANUAL DETALLADO CON INSTRUCCIONES COMPLETAS SOBRE EL FUNCIONAMIENTO INTEGRAL DEL ROBOT.
- LA GUI DEBE CONTAR CON BOTONES ESPECIFICOS PARA CADA FUNCION DEL ROBOT.

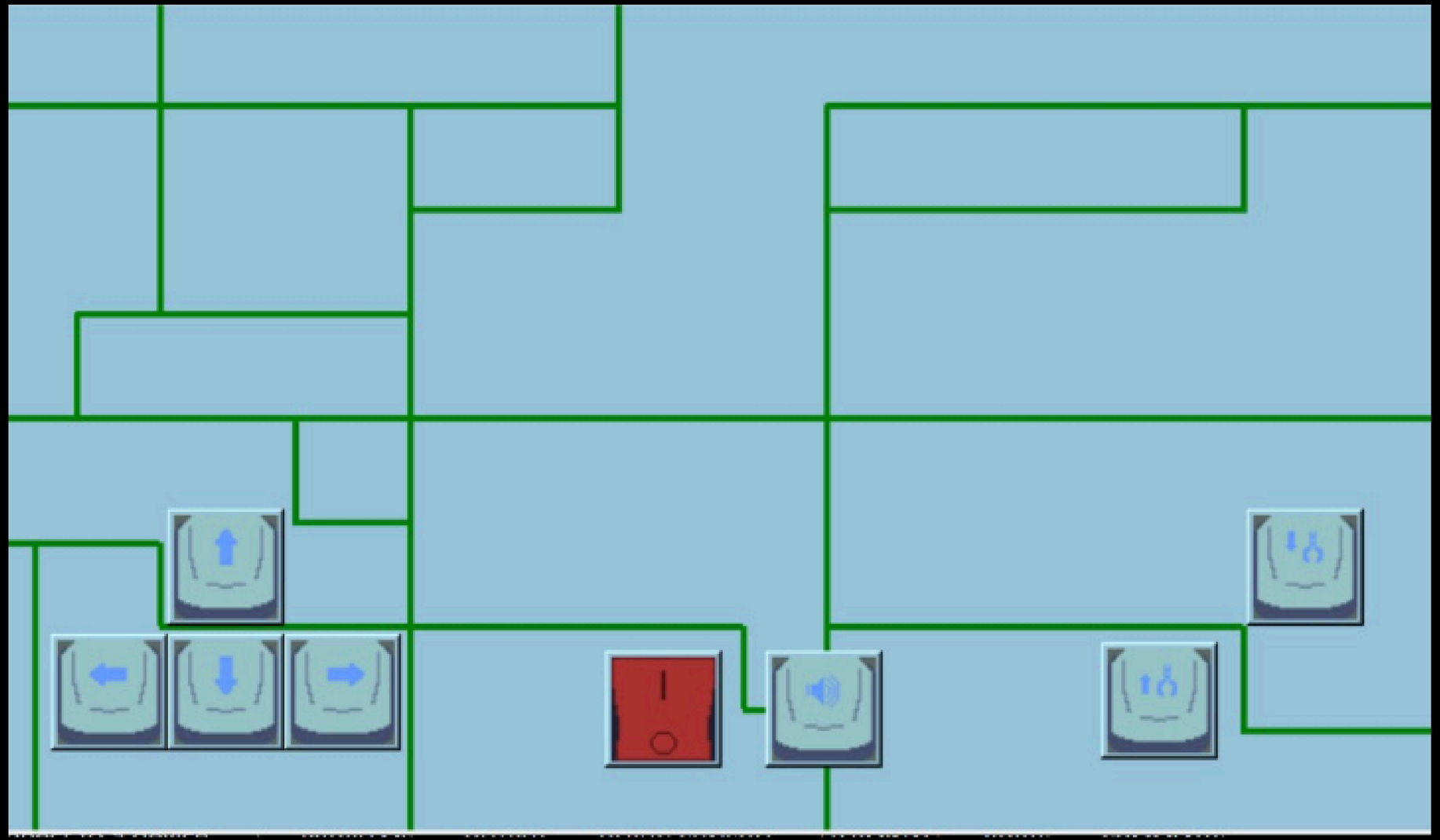
ARQUITECTURA





- CONEXIÓN REMOTA A TRAVÉS DE UN SERVIDOR.
- INPUT DESDE UNA GUI.

INTERFAZ DE USUARIO



TKINTER: PYTHON

2. IMPLEMENTACIÓN

SERVER

Server.py > ...

```python
#!/usr/bin/env python3
import socket
from librari import *

s = socket.socket()
print("Socket creado")
port = 8080
s.bind(('', port))
print("El socket se creo con puerto:{}".format(port))
s.listen(5)
print("EL socket is listening....")
connect, addr = s.accept()
print("Se conecto a {}".format(addr))
conectado()

while True:
    rawByte = connect.recv(1)
    char = rawByte.decode('utf-8')
    if (char == 'w'):
        moveUp()
    if (char == 's'):
        moveDown()

    if (char == 'd'):
        moveRight()

    if (char == 'a'):
        moveLeft()

    if (char == 'g'):
        print("Terminando la sesion....")
        break

    if (char == 'p'):
        para()

    if(char == 'q'):
        levantarGarra()

    if(char == 'r'):
        bajarGarra()

    if(char == 'b'):
        bocina()
apagado()
```

LiBRERÍA

```python
from ev3dev2.sound import Sound
from time import sleep
                  ------

ma = LargeMotor(OUTPUT_B)
mb = LargeMotor(OUTPUT_C)
mg = MediumMotor(OUTPUT_A)
tankmoves = MoveTank(OUTPUT_C,OUTPUT_B)
sound = Sound()


Codeium: Refactor | Explain | Generate Docstring | ×
def moveUp():
    print("Moving up...")
    tankmoves.on(50,50)
Codeium: Refactor | Explain | Generate Docstring | ×
def moveDown():
    print("Moving down...")
    tankmoves.on(-50,-50)
Codeium: Refactor | Explain | Generate Docstring | ×
def moveRight():
    print("Moving right...")
    tankmoves.on(-5,50)
Codeium: Refactor | Explain | Generate Docstring | ×
def moveLeft():
    print("Moving left...")
    tankmoves.on(50,-5)
```

```python
Codeium: Refactor | Explain | Generate Docstring | ×
def levantarGarra():
    print("Moving garra...")
    mg.on_for_seconds(5,1)
Codeium: Refactor | Explain | Generate Docstring | ×
def bajarGarra():
    print("Moving garra...")
    mg.on_for_seconds(-5,1)
Codeium: Refactor | Explain | Generate Docstring | ×
def bocina():
    print("Bocina...")
    sound.beep()
Codeium: Refactor | Explain | Generate Docstring | ×
def para():
    print("Stopping...")
    tankmoves.on(0,0)
Codeium: Refactor | Explain | Generate Docstring | ×
def conectado():
    print("Conectado")
    sound.speak("on")
    sound.beep()
Codeium: Refactor | Explain | Generate Docstring | ×
def apagado():
    print("Apagado")
    sound.speak("off")
    sound.beep()
```

INTERFAZ DE USUARIO

```python
def send_command(command):
    try:
        client_socket.send(command.encode())
        print(f"Comando enviado: {command}")
    except Exception as e:
        print(f"Error al enviar comando: {e}")
```

Codeium: Refactor | Explain | Generate Docstring | ×

```python
def coneccion():
            global g
            if g== False:
              try:
                  print("paso")
                  client_socket.connect(('192.168.71.22', 8080))
                  print("Conectado al servidor.")
                  g = True
                  print("Encender")
                  btn_g.config(image=img_On)
              except Exception as e:
                  print("fallo es posible que su servidor no est
                  print(f"Error al conectar: {e}")
                  g = False

            else:
              print("True")
              send_command("g")
              g = False
              print("apagar")
              client_socket.close()
              btn_g.config(image=img_Off)
```

Codeium: Refactor | Explain | Generate Docstring | ×

```python
def on_key_press(event):
    global client_socket
    global g
    key = event.char
    if key in keys_pressed :
        keys_pressed[key] = True  # Actualizar el estado a pre
        # Manejo de botones visuales...
        print(key)
        print("key es true")
        if key == "w":
            btn_w.config(image=img_up_pressed)
        elif key == "a":
            btn_a.config(image=img_left_pressed)
        elif key == "s":
            btn_s.config(image=img_down_pressed)
        elif key == "d":
            btn_d.config(image=img_right_pressed)
        elif key == "q":
            btn_q.config(image=img_Up_garra_pressed)
        elif key == "r":
            btn_r.config(image=img_down_garra_pressed)
        elif key == "b":
            btn_b.config(image=img_bocina_pressed)

```

INTERFAZ DE USUARIO

```
86  def on_key_release(event):
87
88      key = event.char
89      if key in keys_pressed :
90          keys_pressed[key] = False  # Actualizar el estado a r
91          # Manejo de botones visuales...
92          send_command('p')
93          if key == "w":
94              btn_w.config(image=img_up)
95          elif key == "a":
96              btn_a.config(image=img_left)
97          elif key == "s":
98              btn_s.config(image=img_down)
99          elif key == "d":
100             btn_d.config(image=img_right)
101         elif key == "q":
102             btn_q.config(image=img_Up_garra)
103             send_command('q')
104         elif key == "r":
105             btn_r.config(image=img_down_garra)
106             send_command('r')
107         elif key == "b":
108             btn_b.config(image=img_bocina)
109             send_command('r')
```

```
111
112
    Codeium: Refactor | Explain | Generate Docstring | ×
113 def send_command_continuously():
114     for key, pressed in keys_pressed.items():
115         if pressed and key not in keys_pressed_once:
116             send_command(key)  # Enviar comando al EV3
117             print(f"Enviando comando: {key}")
118
119     root.after(15, send_command_continuously)  # Llama esta f
120
```

3. RESULTADOS

ESTADO ACTUAL DEL PROYECTO

- YA SE HA GENERADO UN DISEÑO SATISFACTORIO PARA EL ROBOT Y SU GARRA.
- EL ROBOT ES TOTALMENTE CAPAZ DE MOVERSE A TIEMPO REAL CON EL TECLADO A TRAVÉS DE UN SERVIDOR Y LIBRERÍAS CREADOS EN PYTHON.
- ACTUALIZACIÓN DE LA WIKI HASTA LA ACTUALIDAD DEL PROYECTO.
- SE HA CREADO UNA INTERFAZ GRÁFICA CON DISEÑOS PROPIOS.
- SE GENERARON BITÁCORAS DURANTE LA DURACIÓN DE TODO EL PROYECTO.

PROBLEMAS ENCONTRADOS

PROBLEMA

- EL ROBOT NO SE MOVÍA DE MANERA FLUIDA Y A TIEMPO REAL CON EL TECLADO.
- LA PLATAFORMA DEL ROBOT NO PODÍA CUMPLIR CON EL REQUERIMIENTO DE LA PROBLEMÁTICA.
- LA WIKI NO AVANZABA EN EL TIEMPO ESTABLECIDO.

SOLUCIÓN

- LOS PROGRAMADORES INVESTIGARON Y BUSCARON LA LÓGICA NECESARIA PARA EL CORRECTO MOVIMIENTO DEL ROBOT.
- EL CONSTRUCTOR SE ENCARGÓ DE DISEÑAR UNA GARRA MÁS ADECUADA.
- LOS DOCUMENTADORES REPARTIERON SU TRABAJO PARA DAR SUFICIENTE PRIORIDAD A LA WIKI.

CONCLUSIÓN