El Desafío del Gato Extremo

Proyecto de Arquitectura de Software

Integrantes

José Gabriel Rojas Pérez

José Ángel Lozano Cruz

Giovani Jimenez Bonilla

Materia

Arquitectura de Software

Docente

Alfredo García Suárez

Introducción al Proyecto

Desarrollo de un juego interactivo de "Tres en Raya" para fomentar el aprendizaje en primaria mediante gamificación y un mando físico.

Interacción Humano-Computadora

Juego clásico "Tres en Raya" adaptado para interacción con mando físico.

Arquitectura Cliente-Servidor

Backend local para preguntas en tiempo real vía HTTP, facilitando la carga de contenido educativo.

Mando Físico

Joystick conectado a ESP32/Arduino para controlar el juego, transformando movimientos en datos.

Objetivo y Alcance

Diseñar una plataforma HCI que mejore la experiencia del usuario y fomente el aprendizaje dinámico a través del juego del gato.

Objetivo Principal

Mejorar la interacción y el aprendizaje con un juego del gato, utilizando un mando físico y una interfaz dinámica.

Alcance del Sistema

Dos jugadores compiten en un tablero 3x3, controlado por ESP32. La lógica del juego y el procesamiento de jugadas se ejecutan en Python.

Dinámica Educativa del Juego

El clásico "Tres en Raya" se transforma en una herramienta educativa donde cada movimiento requiere conocimiento.

01

Pregunta Obligatoria

Antes de cada movimiento, el jugador debe responder una pregunta aleatoria.

02

Ficha por Respuesta Correcta

Si la respuesta es correcta, el jugador coloca su ficha (X o O).

03

Pérdida de Turno

Si la respuesta es incorrecta, el jugador pierde su turno.

04

Banco de Preguntas

Mínimo 20 preguntas de nivel primaria, con opciones de modo vs. IA y temas personalizables.

Arquitectura Cliente-Servidor

El sistema centraliza la lógica del juego y la gestión de contenido educativo en el servidor, mientras el cliente maneja la interacción.

1

Backend con Flask (Python)

Núcleo del servidor, gestiona preguntas y entrada del joystick.

2

Frontend con HTML/JS

Interfaz de usuario, actualiza cursor y activa preguntas.

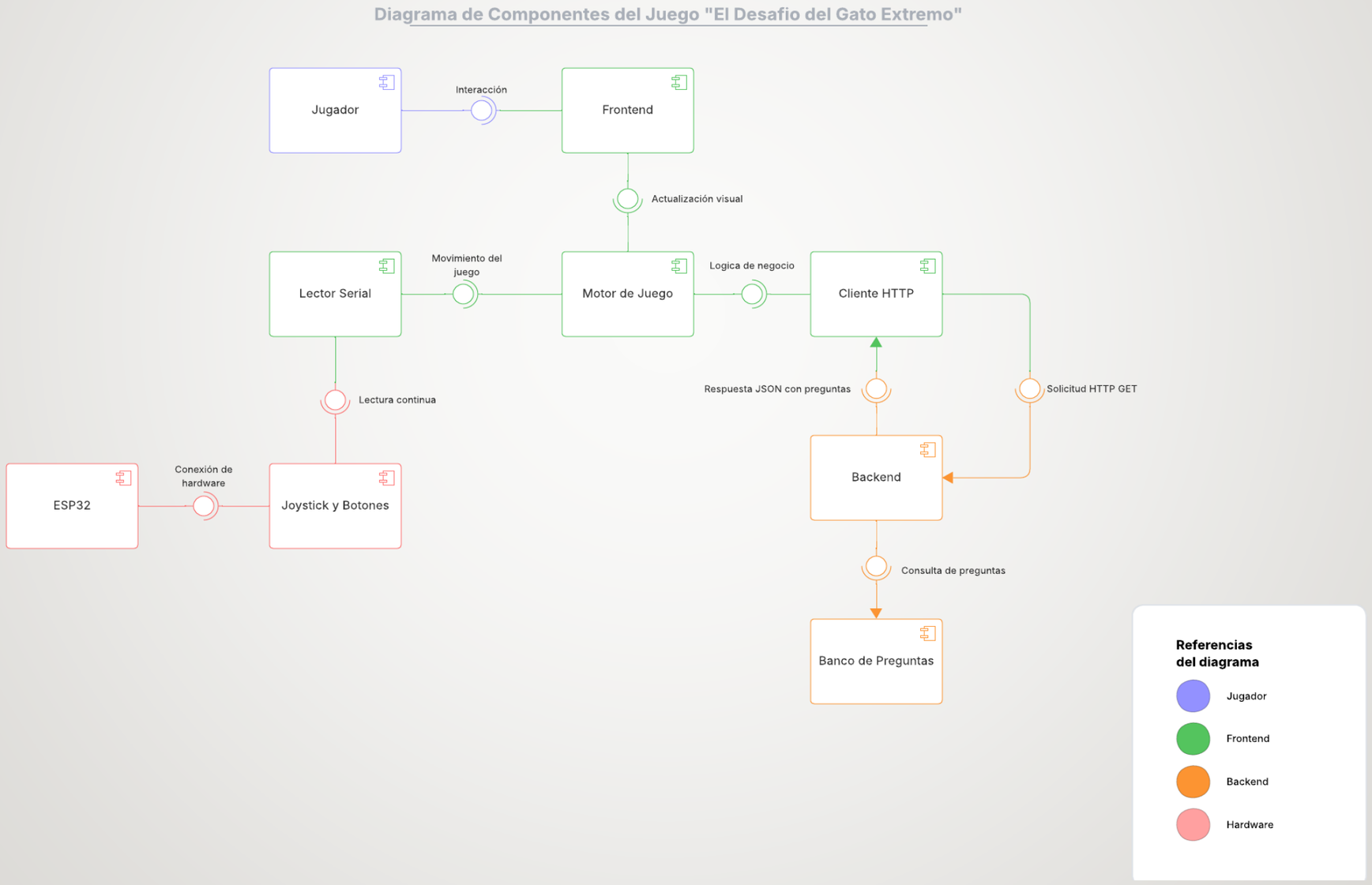
3

Servicio Serial Asíncrono

Comunicación con ESP32 en hilo separado para evitar bloqueos.

Diagrama de Componentes

Arquitectura del Juego en base a sus componentes principales.



Especificaciones Técnicas

Detalles de la implementación del banco de preguntas y la integración de la interacción humano-computadora.

Banco de Preguntas

- JSON con preguntas, opciones, respuesta correcta y explicación.
- 36 preguntas de temas mixtos (matemáticas, geografía, historia, lenguaje).
- Tiempo límite de 15 segundos para respuestas rápidas.

Integración HCI y Estado

- Modelo de datos del mando: "x", "y", "sw", "btn", "connected".
- Manejo de errores y reconexión para tolerar desconexiones.

Requerimientos del Proyecto

Identificación de las funcionalidades clave y criterios de calidad para el desarrollo del juego.

- ## Modos de Juego

Jugador vs. Jugador y Jugador vs. IA (opcional).

- ## Navegación y Preguntas

Mando ESP32 para cursor, preguntas educativas antes de cada jugada.

- ## Validación y Puntuación

Backend valida jugadas, detecta victoria/empate, asigna puntos extra por rapidez.

- ## Usabilidad y Confiabilidad

GUI legible, flujo sencillo, tolerancia a desconexiones USB.

Implementación de Componentes

La viabilidad técnica se demuestra con la integración de hardware y software heterogéneos.

Generador de Preguntas	Función questions() en app.py y preguntas.json
Comunicación Serial	Hilo serial_worker con biblioteca serial y threading
Motor de Interfaz	index.html y game.js para la vista principal
Cliente de Input	Función get_input() en app.py

Código Fuente y Evidencias

El desarrollo del control y la interfaz del juego muestran la integración de hardware y software.

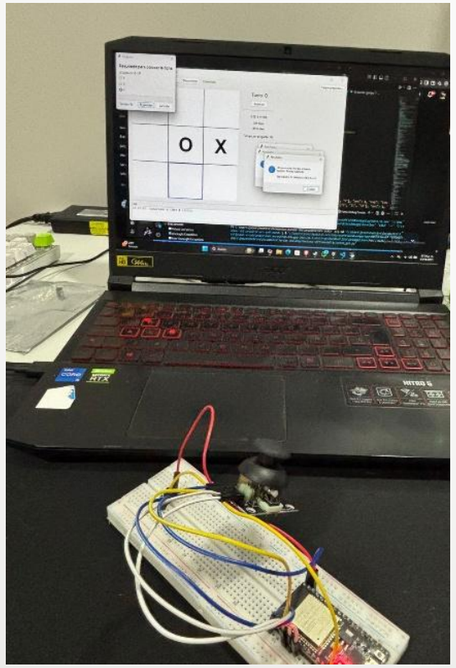
Control Desarrollado

ESP32 con joystick y botones, pines definidos para ejes y pulsadores.

```python
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/get_input')
def get_input():
    with serial_lock:
        return jsonify(joystick_state)

@app.route('/questions')
def questions():
    limit = int(request.args.get("limit", 35))
    if not os.path.exists(DATA_PATH): return jsonify({"items": []})
    with open(DATA_PATH, "r", encoding="utf-8") as f:
        data = json.load(f)
    items = data.get("items", [])
    random.shuffle(items)
    return jsonify({"items": items[:limit]})

if __name__ == "__main__":
    # use_reloader=False evita el error de "Permiso denegado" en el puerto COM
    app.run(host="0.0.0.0", port=5000, debug=True, use_reloader=False)
```

Circuito del Control

Conexiones físicas del joystick y botones al microcontrolador.



Código del Control

Lectura de posición (x, y) y estado de botones (sw, btn) con formato de salida serial.

```python
# Estado global del Joystick
joystick_state = {"x": 2048, "y": 2048, "sw": 0, "btn": 0, "connected": False}
serial_lock = threading.Lock()
```

```python
def serial_worker():
    """Hilo que lee el Arduino en segundo plano"""
    global joystick_state
    ser = None

    while True:
        if ser is None:
            try:
                ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0.1)
                print(f"✅ Conectado a {SERIAL_PORT}")
                time.sleep(2)
                ser.write(b'L') # Encender LED
                with serial_lock:
                    joystick_state["connected"] = True
            except Exception:
                with serial_lock:
                    joystick_state["connected"] = False
                time.sleep(2)
                continue
```

Conclusión del Proyecto

El código del control es fundamental, interpretando las acciones del jugador para una experiencia intuitiva y entretenida.

El programa convierte acciones simples como mover la palanca o presionar un botón en decisiones dentro del juego, haciendo posible la sincronización entre el microcontrolador y el software de escritorio.

Un módulo pequeño, bien diseñado y documentado, puede tener un impacto directo en la calidad y la sensación final de todo el proyecto interactivo.