Sistema de control táctil para videojuego

DINO utilizando Arduino AQUILES

Integrantes : Alma Carena , Mateo Lugo , Facundo Noriega , Santino Trevisano y Dante Bassus

Institucion : Tecnica N°1 de Vicente López

Curso : Laboratorio de Programación 5°3 A-B y

Profesores: Mansilla Muñoz York E. / Ganduglia Yamil / Salimbeli

Fecha: 00/00/2025

Índice

1. Introducción

El programa que desarrollamos trata sobre un dino corriendo infinitamente mientras esquiva obstáculos, integramos diferentes librerías para realizar el juego y implementamos Mysql y Arduino para añadir mas mecanicas y un mando físico.

1.1 Justificación del proyecto

El propósito de este trabajo fue trasladar la experiencia clásica del videojuego del dinosaurio a un entorno físico, incorporando elementos tangibles que permitan al usuario interactuar de forma más directa.
Se eligió Arduino UNO por su facilidad de uso y compatibilidad con Python mediante la librería PySerial, lo que posibilita una comunicación fluida entre el hardware y el software.
Además, se buscó desarrollar habilidades de trabajo en equipo, diseño digital, y programación modular, fomentando la creatividad en el uso de tecnologías accesibles.

1.2 Objetivos

Desarrollar un videojuego controlado mediante sensores táctiles utilizando Arduino y Python, integrando aspectos de programación, diseño visual y electrónica.

1.3 Material Articuladas

Para el proyecto articulamos dos materias, en las cuales utilizamos los conocimiento adquiridos para realizar y pulir aun mas el proyecto, implementamos una materia de taller y una curricular las cuales son las siguientes:

1.3.1 Inglés

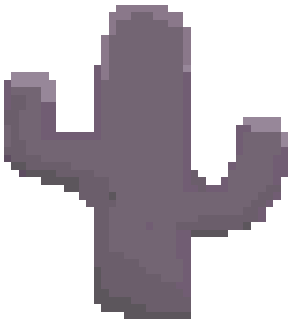
En la materia de Inglés trabajamos sobre la traducción y presentación internacional del proyecto, elaborando tanto el manual del usuario como las instrucciones del juego en este idioma. De esta forma, pudimos integrar el aspecto técnico con la comunicación en inglés, permitiendo que el proyecto sea más accesible y entendible para un público más amplio.

1.3.2 Sistemas Digitales

En la materia de Sistemas Digitales nos enfocamos en la parte electrónica y lógica del proyecto, implementando el Arduino como componente principal del mando. A través de los conocimientos sobre circuitos, sensores y programación digital, logramos diseñar el funcionamiento del control, los *touch* y la lógica que determina el encendido del LED y las respuestas del sistema.

2. Diseño y Animaciones

Los bocetos y el paso a paso de los frames, entorno y diseños del proyecto fueron hechos en la plataforma Pixilart. Cuando veíamos que no podíamos solucionar o arreglar un problema de dibujo le pedíamos a una IA que retoque el dibujo como el siguiente caso

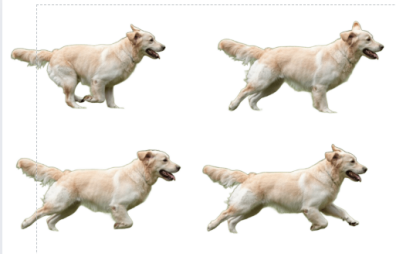
Nosotros	**IA**
	

Imágenes de los frames : Después de terminar con los bocetos iniciales de los cactus, pensamos que sería más original y llamativo romper con la estética pixelada del juego. Para eso decidimos reemplazar al cactus por un perro con estilo realista, generando un contraste con el resto de los elementos.
En grupo investigamos diferentes razas y estilos de perros hasta ponernos de acuerdo en cuál sería nuestro personaje principal. Una vez elegido, utilizamos Gemini para generar los distintos frames de animación necesarios:

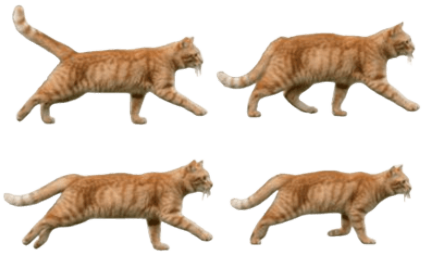
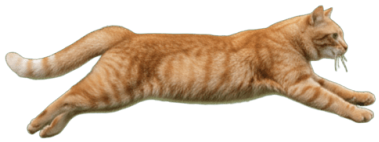
- Algunos frames corriendo
- Otros frames saltando.
- Y finalmente, un frame en el aire.

De esta forma conseguimos que el personaje mantenga una consistencia en la animación (Que era lo que buscábamos) y al mismo tiempo destacándose dentro de la interfaz del juego

2.1 Frames del perro

Frames Corriendo	Frames Saltando	Frames Perro en el aire
		

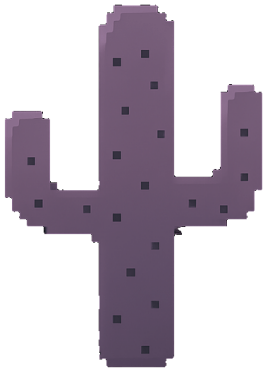
2.2 Frames del gato

Frames Corriendo	Frame Saltando	Frame en el aire
		

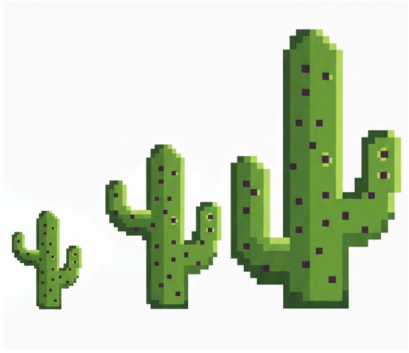
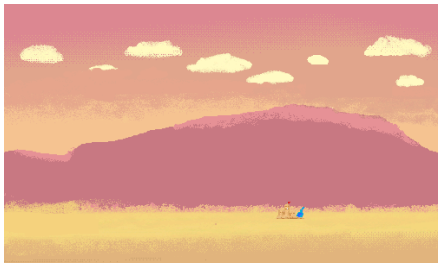
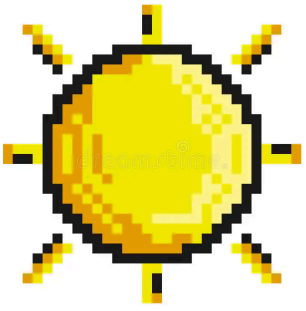
2.3 Frames del ave

	Otros frames	
		

2.4 Imágenes que acompañan al Mundo1

Fondo	Cactus	Luna
		

2.5 Imágenes que acompañan al Mundo2

Cactus	Fondo	Sol
		

3. Código

Una vez finalizado el proceso de diseño de las imágenes y la generación de los distintos frames del personaje principal, procedimos a integrarlos dentro de la estructura del juego. Para ello, recurrimos al apoyo de ChatGPT, herramienta con la cual desarrollamos la base de la programación, incluyendo la lógica de movimiento, las colisiones y la organización general de la interfaz.

Sobre esta base incorporamos las imágenes previamente creadas, en especial los distintos estados del perro (corriendo, saltando y en el aire). Este paso resultó ser muy importante, ya que permitió pasar de un conjunto de elementos aislados a un modelo funcional en el que las animaciones y la mecánica comenzaron a trabajar de manera conjunta.

3.1 Archivos Utilizados

Game_object.py : El archivo game_objects.py contiene la definición de las clases que representan los distintos elementos del juego. En este caso, se encuentran clases como Perro, Obstáculo y Fondo. Cada clase posee atributos y métodos que encapsulan su comportamiento: el perro controla la animación y el salto, los obstáculos gestionan su desplazamiento y eliminación al salir de la pantalla, y el fondo administra el desplazamiento continuo que genera la sensación de movimiento.

main.py : El archivo main.py cumple el rol de punto de entrada del juego. Allí se configura la ventana principal, se cargan los recursos gráficos, se inicializan los objetos principales (como el personaje, el fondo y los obstáculos) y se ejecuta el bucle principal. Dicho bucle es el núcleo del juego, ya que gestiona la captura de eventos (como las teclas presionadas), la actualización de la lógica de los objetos (movimiento, animaciones y detección de colisiones) y la representación visual de todos los elementos en la pantalla.

utils.py : el archivo utils.py se utiliza como un conjunto de herramientas auxiliares. En él se definen funciones de uso común, como la que permite mostrar texto en pantalla. Estas funciones pueden emplearse en distintas partes del proyecto, evitando la repetición de código y contribuyendo a la reutilización y legibilidad del programa.

menu.py : El archivo *menu.py* contiene la lógica y el diseño del menú principal del juego. Desde aquí el jugador puede acceder a las distintas opciones, como iniciar partida, seleccionar personaje, mundo o sonido, y salir del juego. Su función principal es gestionar la navegación entre las diferentes pantallas del programa.

seleccion_personaje.py : En *seleccion_personaje.py* se define la pantalla donde el jugador elige el personaje que va a utilizar. Se muestran las distintas opciones disponibles y se guarda la selección para que el juego la utilice al comenzar la partida.

seleccion_mundo.py : El archivo *seleccion_mundo.py* permite al jugador elegir el entorno o escenario en el que se desarrollará el juego. Cada mundo puede tener un fondo o dificultad diferente, y la elección se aplica al iniciar el juego.

elegir_nombre.py : El archivo elegir_nombre.py gestiona la pantalla donde el jugador debe ingresar su nombre antes de comenzar la partida. Se encarga de mostrar el campo de texto, validar la entrada y devolver el nombre seleccionado, que luego será utilizado para registrar el puntaje o mostrarlo en el ranking.

mostrar_ranking.py : En mostrar_ranking.py se encuentra la lógica para desplegar la tabla de posiciones o ranking de jugadores. Este archivo se encarga de leer los puntajes almacenados, ordenarlos y presentarlos en pantalla de manera clara, permitiendo al jugador ver su posición respecto a los demás.

seleccionar_sonido.py : El archivo seleccionar_sonido.py permite al jugador elegir la música o los efectos de sonido que acompañarán la partida. Gestiona la lista de pistas disponibles y aplica la selección para personalizar la experiencia de juego.

Transiciones.py: El archivo trancisiones.py contiene los elementos, formatos y valores de las transiciones del menú y del juego, algunas no se usaron pero están guardadas ahí, cuando son llamadas dependiendo de qué transición sea se ejecuta.

3.2 Librerías Utilizadas

Pygame : Es la librería principal que usamos para crear el juego. Gracias a Pygame pudimos manejar los gráficos, los sonidos y todo lo que el jugador hace con el teclado o el mouse. Básicamente, es la que nos permitió mostrar el personaje, los obstáculos y el fondo en pantalla.

Pyserial : Nos sirvió para conectar el juego hecho en Python con el Arduino. Con esta librería logramos que ambos se comuniquen por el puerto serial, así el Arduino puede enviar señales al juego o recibir información de él. Por ejemplo, sirve para que el personaje salte cuando tocamos un sensor físico.

Random : Se usa para generar números al azar. Con esto pudimos hacer que el juego no sea siempre igual, ya que permite que los obstáculos aparezcan en distintos momentos o posiciones. Así el juego se vuelve más variado y divertido.

Sys : Esta librería ayuda a que el programa se comunique con la computadora. En nuestro caso, la usamos para cerrar el juego de forma segura cuando termina o el jugador sale.

Sys : Esta librería ayuda a que el programa se comunique con la computadora. En nuestro caso, la usamos para cerrar el juego de forma segura cuando termina o el jugador sale.

Os : Nos permite trabajar con archivos y carpetas dentro de la computadora. La usamos para poder encontrar fácilmente las imágenes o sonidos que usa el juego, sin importar en qué carpeta esté guardado el proyecto.

Time : La usamos para medir el tiempo y controlar la velocidad de ciertas acciones, como la espera para la conexión con Arduino o los intervalos entre eventos del juego.

Mysql.connector : Esta librería nos permitió conectar el juego con una base de datos MySQL. Así pudimos guardar y recuperar los puntajes de los jugadores, mostrando un ranking actualizado.

Serial.tools.list_ports : Es un submódulo de **pyserial** que nos ayuda a listar todos los puertos seriales disponibles en la computadora. Lo usamos para facilitar la detección del Arduino cuando está conectado.

Pymysql : Se utiliza para la conexión y manejo de la base de datos MySQL desde Python, permitiendo ejecutar consultas y gestionar los datos de los usuarios y puntajes.

3.3 Consultar código y ver los avances :

Consulta en el siguiente enlace para ver el Código

4. Manual del Programador

Consulta en el siguiente enlace para ver el Manual del Programador

5. Implementación de Base de datos

El módulo “Ranking” permite almacenar y mostrar los puntajes de los jugadores. Cada vez que un jugador finaliza una partida, su puntaje se guarda automáticamente en la base de datos “Supermercado” y se actualiza si logra superar su récord anterior. Con esta información, el módulo genera un Top 3 de mejores puntajes, mostrando claramente a los jugadores con mejor rendimiento y fomentando la competencia y la superación dentro del juego.

5.1. Estructura de la base de datos

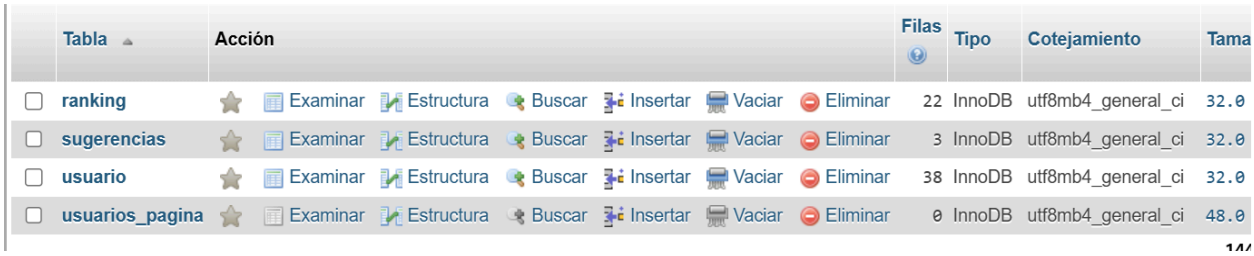
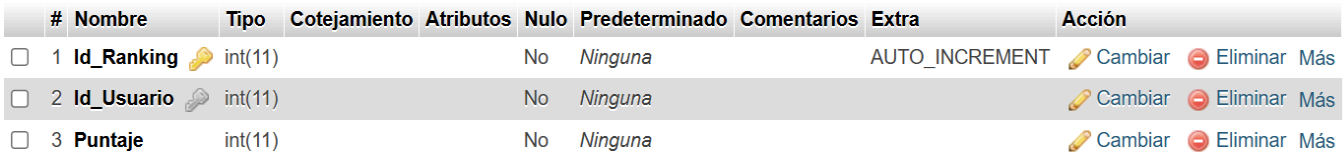


	Tabla	Acción	Filas	Tipo	Cotejamiento	Tama
☐	ranking	Examinar Estructura Buscar Insertar Vaciar Eliminar	22	InnoDB	utf8mb4_general_ci	32.0
☐	sugerencias	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0
☐	usuario	Examinar Estructura Buscar Insertar Vaciar Eliminar	38	InnoDB	utf8mb4_general_ci	32.0
☐	usuarios_pagina	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	48.0

5.2. Estructura tabla “ranking”



	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	**Id_Ranking**	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
☐	2	**Id_Usuario**	int(11)			No	*Ninguna*			Cambiar Eliminar Más
☐	3	**Puntaje**	int(11)			No	*Ninguna*			Cambiar Eliminar Más

5.2.1 Claves foráneas y principales en tabla “ranking”

Id_Ranking : Clave primaria

Id_Usuario : Clave foránea de la tabla “usuario”

5.4. Estructura tabla “usuario”

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	**Id_Usuario**	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	**Nombre**	text	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más

5.4.1 Claves principales en la tabla “usuario”

Id_Usuario : Clave principal

5.5. Estructura tabla “usuarios_pagina”

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	**Id_UsuarioPagina**	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambia
2	**NomUsuario**	varchar(100)	utf8mb4_general_ci		No	*Ninguna*			Cambia
3	**email**	varchar(255)	utf8mb4_general_ci		No	*Ninguna*			Cambia
4	**password**	varchar(255)	utf8mb4_general_ci		No	*Ninguna*			Cambia
5	**active**	tinyint(1)			Sí	1			Cambia
6	**fecha_registro**	timestamp			No	current_timestamp()			Cambia

5.5.1 Claves principales en la tabla “usuarios_pagina”

Id_UsuarioPagina : Clave primaria

NomUsuario : Clave foránea

email : Clave foránea

5.6. Estructura tabla “sugerencias”

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
☐	1	**Id_Sugerencia**	int(11)			No	*Ninguna*		AUTO_INCREMENT
☐	2	**Id_UsuarioPagina**	int(11)			No	*Ninguna*		
☐	3	**titulo**	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
☐	4	**categoria**	enum('gameplay', 'graphics', 'audio', 'controls', ...	utf8mb4_general_ci		No	*Ninguna*		
☐	5	**descripcion**	text	utf8mb4_general_ci		No	*Ninguna*		
☐	6	**valoracion**	int(1)			Sí	0		
☐	7	**fecha_creacion**	timestamp			No	current_timestamp()		

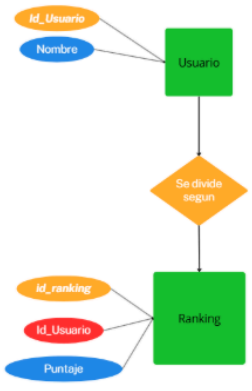
5.6.1 Claves principales en la tabla “sugerencias”

Id_Sugerencia : Clave primaria

Id_UsuarioPagina : Clave foránea del campo “Id_UsuarioPagina” de la tabla “usuarios_pagina

5.7. Diagrama de entidad de relación de la base de datos

Consulta en el siguiente enlace para ver el diagrama entidad relación



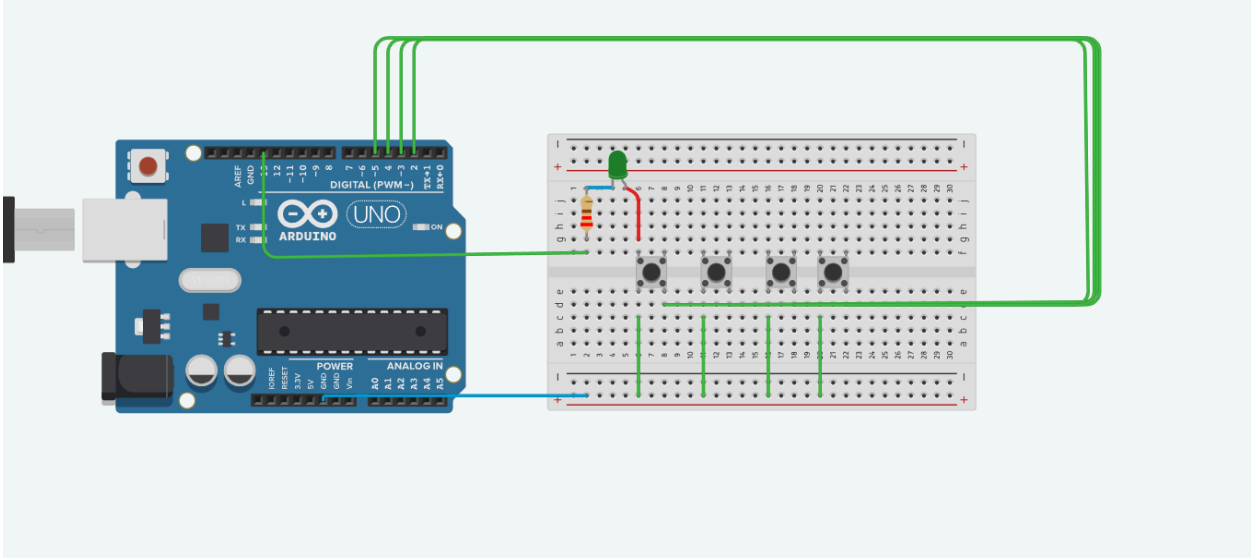
Donde :
Amarillo : Son claves primarias
Rojo : Claves foraneas

6. Arduino “AQUILES”

Nosotros creamos un mando de juego compuesto por cuatro botones principales, cada uno destinado a ejecutar una función específica dentro de los distintos aspectos del juego y sus diferentes módulos, tal como se explicó previamente. Cada botón cumple un papel particular, permitiendo al usuario interactuar con el entorno del juego de manera intuitiva y ordenada.

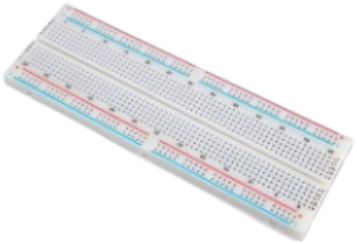
6.1 Circuito diseñado en Tinkercad

Circuito de prueba hecho en Tinkercad completamente funcional.



6.2 Componentes

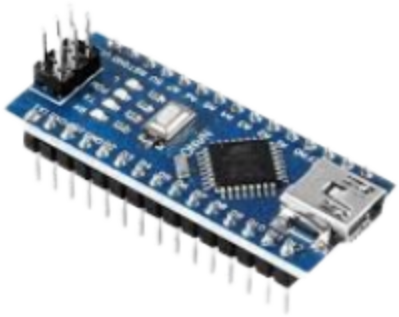
Lista de materiales para fabricar y realizar el mando.



Protoboard: Se utilizó como principal componente, para poder realizar las conexiones con el cableado y los demás componente**s**

Resistencia de 220Ω x1: Se utilizó una resistencia de 220Ω para la luz led.

Botones pulsadores x4: Los botones pulsadores se usaron para poder interactuar con las funciones del juego.



Arduino Nano: El arduino se usó para poder programar con los diferentes pines, los botones y el led, para poder controlar el juego.

Cables: Se usaron para conectar todos los elementos en el protoboard.

Led x1: Se usó un led para poder tener una indicación de que los botones pulsadores funcionan en cada toque.

6.3 Tablas de verdad

En el momento de organizar y especificar qué función cumple cada *touch*, realizamos primero un debate acerca de la cantidad de sensores táctiles que íbamos a utilizar. Luego de analizar las necesidades del sistema y las posibles interacciones dentro del contexto del juego, se decidió trabajar con cuatro touch. Esta cantidad nos permite disponer de un buen equilibrio entre control, respuesta y simplicidad del circuito, además de facilitar la interpretación de las salidas. Cada *touch* representa una entrada independiente del sistema, asociada a una acción o evento específico del juego. Su funcionamiento está diseñado para que el sistema distinga claramente entre una activación individual y una activación múltiple, lo cual resulta fundamental para evitar conflictos en la lógica del programa.

El comportamiento general de las salidas es el siguiente:

1. Cuando se presiona un solo touch, el LED se enciende indicando que el sistema ha reconocido correctamente la entrada. En este caso, la función correspondiente del juego se ejecuta normalmente, generando una salida lógica igual a 1.
2. Por el contrario, cuando se presionan dos o más touch al mismo tiempo, el LED también se enciende (ya que el sistema detecta actividad), pero no se ejecuta ninguna acción dentro del juego, ya que se interpreta como una condición no válida. En este caso, la salida lógica pasa a ser 0.

De esta forma, la función booleana del sistema refleja esta lógica: la salida es 1 únicamente cuando exactamente una entrada está activa, y 0 en todos los demás casos. Esta condición asegura que el juego responda de forma precisa a la interacción del jugador, evitando errores o respuestas no deseadas cuando se presionan varios sensores simultáneamente.

1.

Touch 1	Touch 2	Touch 3	Touch 4	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

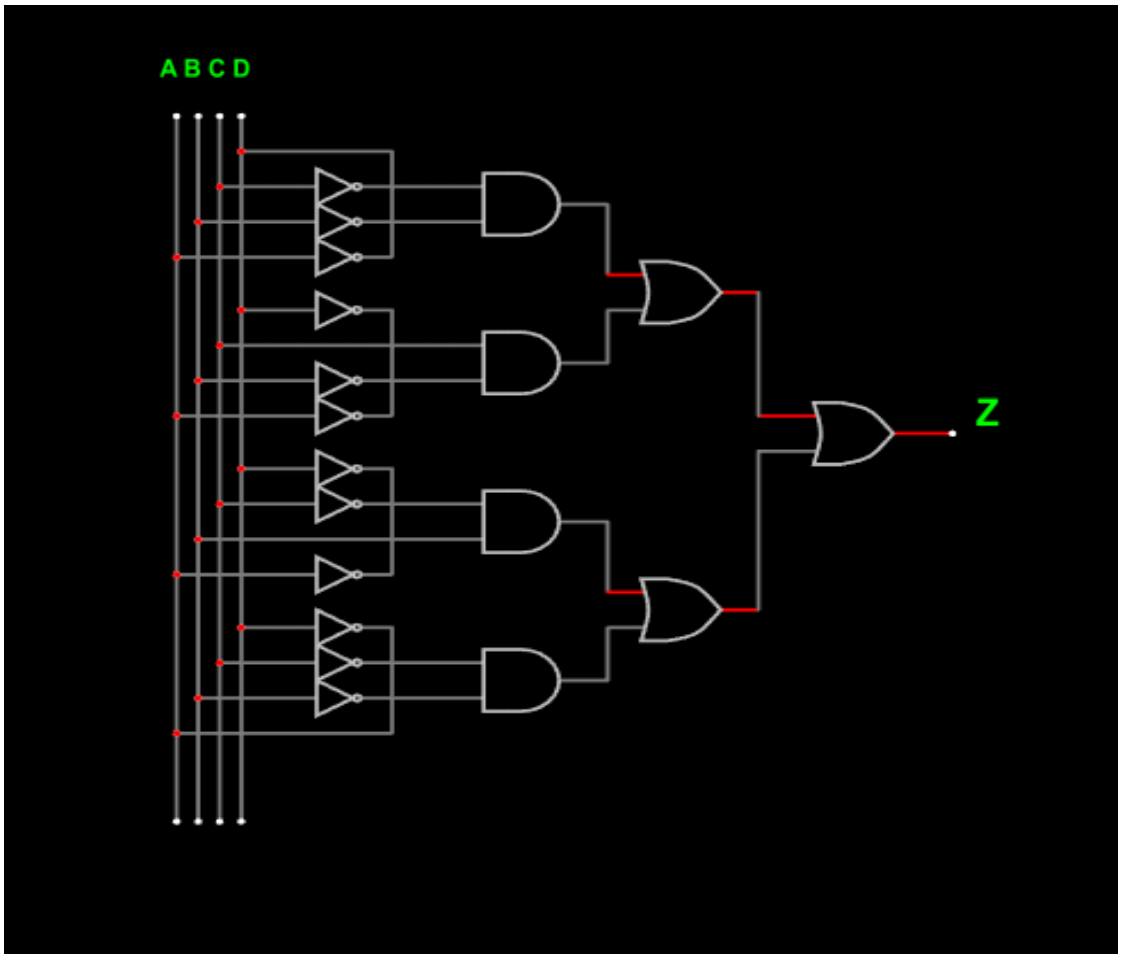
6.4 Álgebra de boole

Al aplicar las leyes del álgebra de Boole a la función obtenida a partir de los minitérminos, se intentó realizar una simplificación adicional. Sin embargo, debido a que los términos no comparten combinaciones que permitan agrupaciones o reducciones, no fue posible simplificar más la expresión. De forma lógica se podría con una compuerta de tres XOR T1⊕T2⊕T3⊕T pero no cumpliria con los requisitos del circuito porque cuando haya tres entradas activas Z va a ser 1, cosa que no sigue el funcionamiento.

Z = A not B not C not D + not A B not C not D + not A not B C not D + not A not B not C D

6.5 Circuito

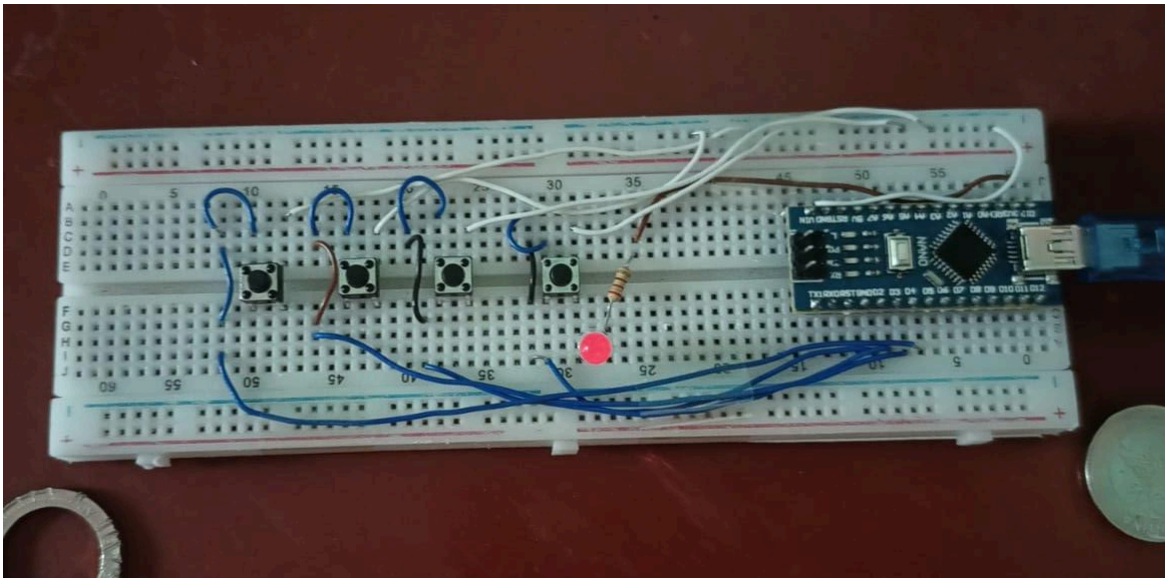
Representación de la estructura del Arduino hecho en: MASTER PLC



5.5 Hardware

En el momento de hacer el controlador HID (Aquiles) optamos por un diseño sencillo, fiel a la forma del protoboard el cual esta pintado de azul para generar contraste con los botones los cuales son rojos y hacer que el microcontrolador no resalte tanto y por ultimo un led el cual es de color fucsia que cuando se prende es llamativo por su color pero no por su brillo , siendo comodo visualmente.

6.5.1. Circuito por dentro



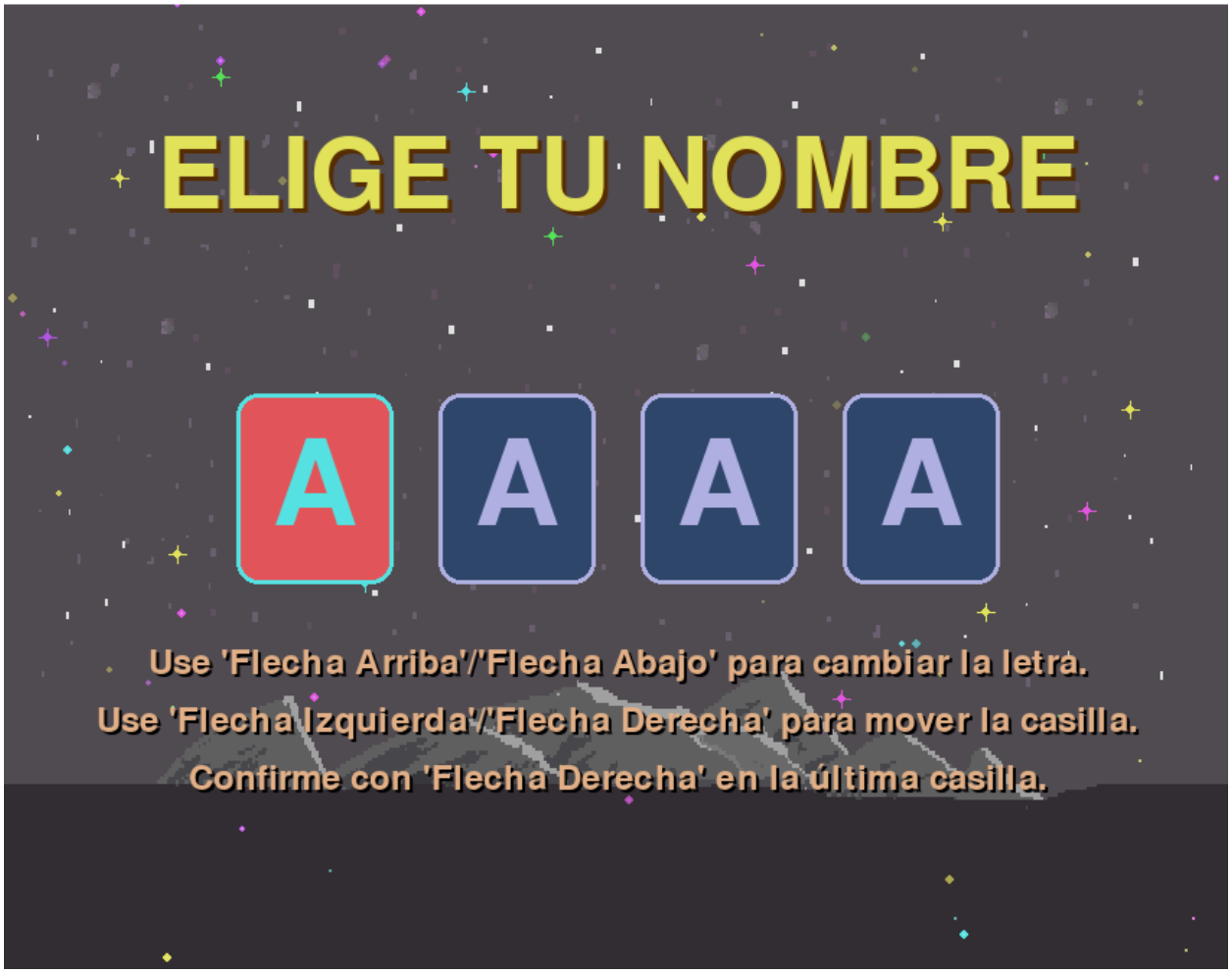
7. Gameplay “DINO”

7.1. Inicio y selección de idioma

Al iniciar “DINO”, el jugador elige el idioma de la interfaz (español o inglés), lo que facilita la accesibilidad y comprensión para distintos usuarios.

7.2. Ingreso de nombre

Luego, el jugador ingresa su nombre, que se usará para guardar puntajes y mostrar en el ranking. Este paso personaliza la experiencia y permite identificar los logros individuales.



7.3. Menú principal

El menú principal presenta varias opciones:

7.3.1 Jugar

Inicia la partida y comienzo el juego

7.3.2 Salir

Cierra el juego.

7.3.3 Ranking

Muestra el top 3 de los records más grandes que hay en la Base de datos

Id_Ranking	Puntaje	Id_Usuario
1	1500	1
2	1200	2
3	1800	3
4	900	4
5	465	9

7.3.4 Elegir personaje

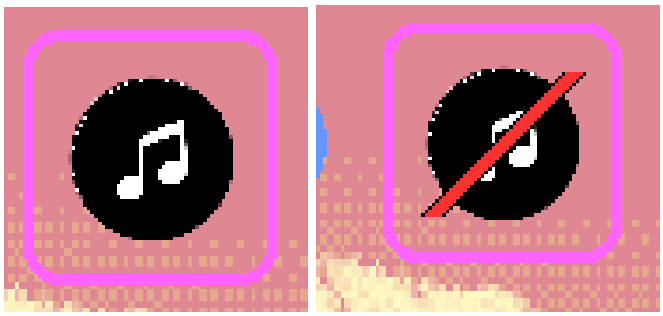
Selecciona entre distintos dinosaurios. (Son solo visuales no alteran el gameplay)

7.3.5 Seleccionar mundo

Elige el escenario donde se desarrollará la partida. (Son solo visuales no alteran el gameplay)

6.3.6 Icono de sonido

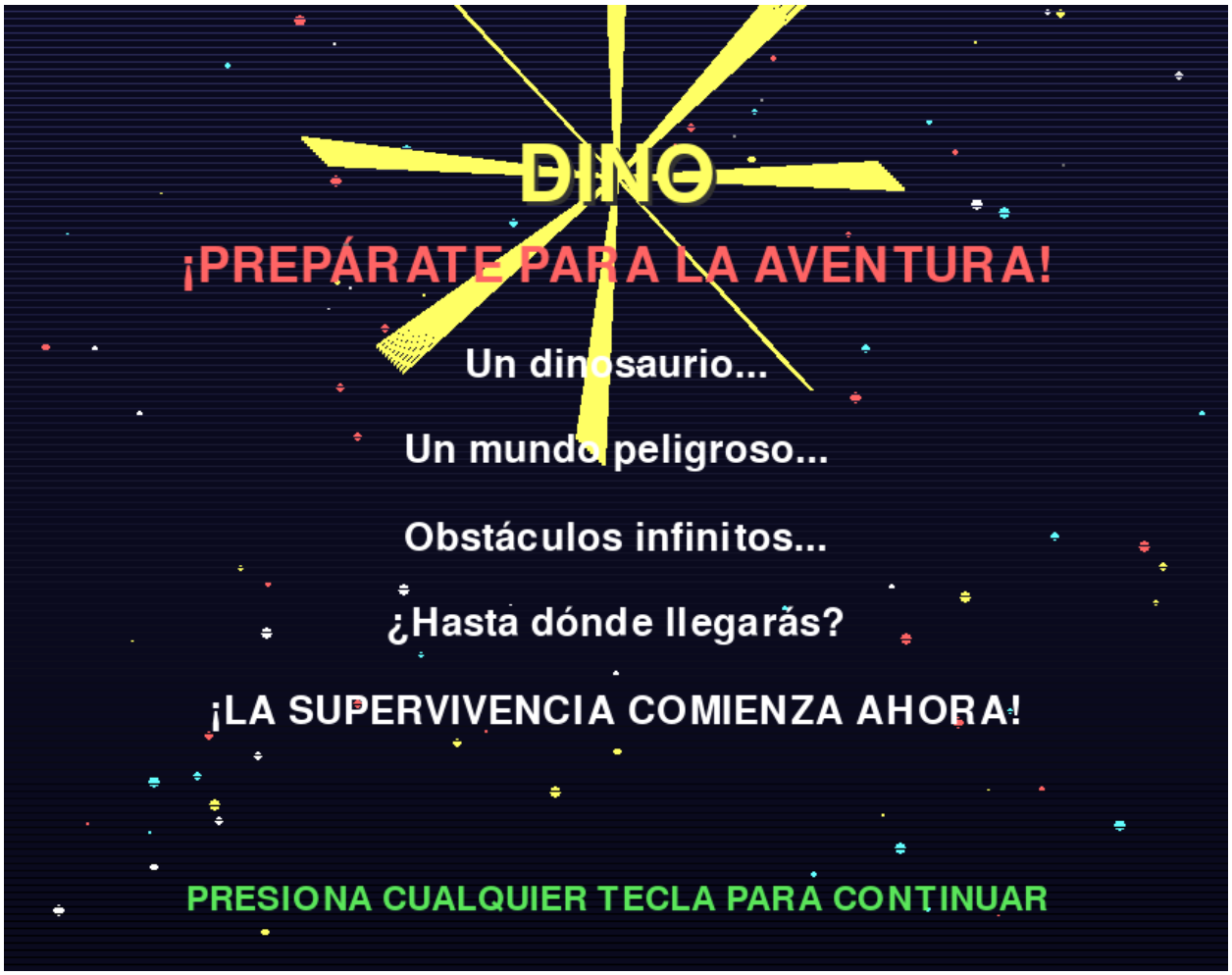
Un botón para silenciar o activar el volumen del juego.



7.4 Comienzo

7.4.1 Cinemática

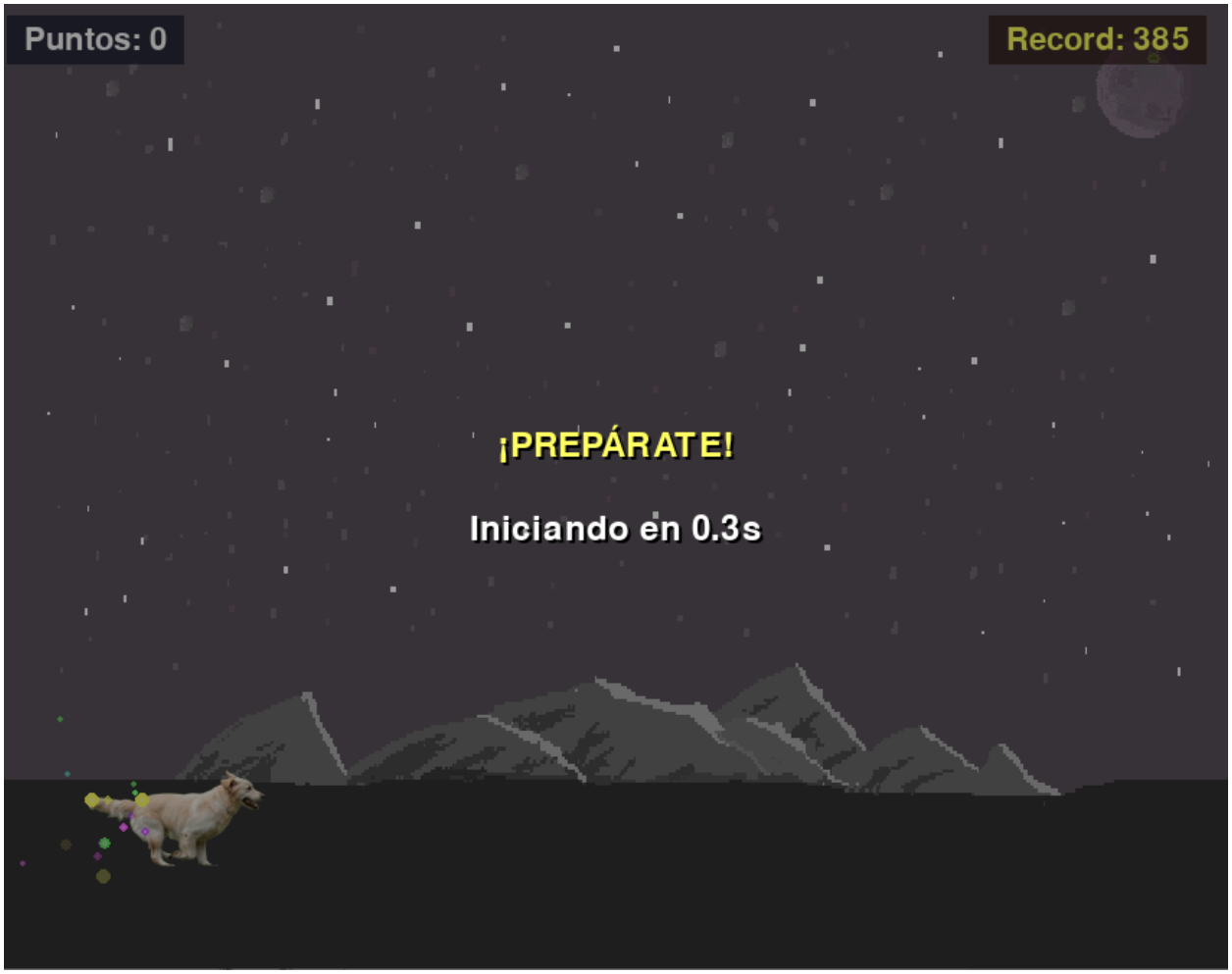
El gameplay de “DINO” comienza con una breve cinemática de introducción, donde se presenta el mundo y el personaje elegido. Esta secuencia inicial incluye música de fondo y animaciones que ambientan al jugador y generan expectativa antes de la partida.



7.4.2 Inicio

Tras la cinemática, hay un tiempo de preparación: el juego muestra una pantalla de “¡Preparados!” o “Ready!” durante unos segundos. Este intervalo permite al jugador concentrarse, familiarizarse con los controles y prepararse mentalmente para la acción.

Una vez finalizado el tiempo de preparación, el juego inicia automáticamente.



7.4.3 Mecanicas

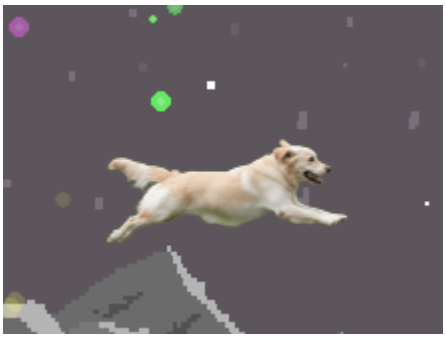
El dinosaurio comienza a correr y el jugador debe reaccionar rápidamente para esquivar obstáculos (cactus, aves, etc.) usando saltos y agachamientos. Las mecánicas principales son:

7.4.3.1 Salto

El jugador presiona una tecla (o botón Arduino) para saltar sobre obstáculos. Entre más la mantenga más largo es el salto.

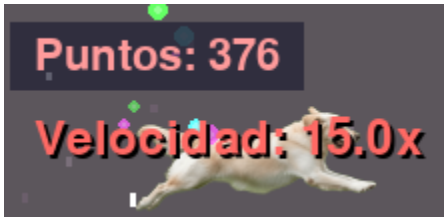
7.4.3.2 Agacharse

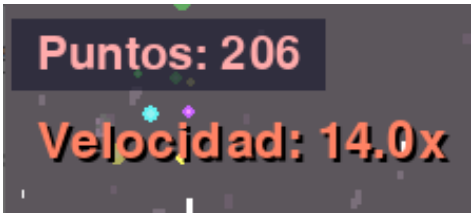
Otra tecla o (o botón del Arduino) permite agacharse y evitar obstáculos bajos. Si se presiona en el Aire el Dino planea.

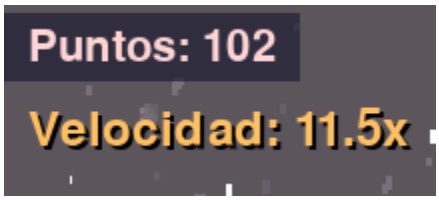


7.4.3.3 Velocidad progresiva

A medida que avanza la partida, la velocidad del juego aumenta, incrementando la dificultad y la exigencia de reflejos.







Aumenta cada 100 puntos hasta un máximo de 50%.

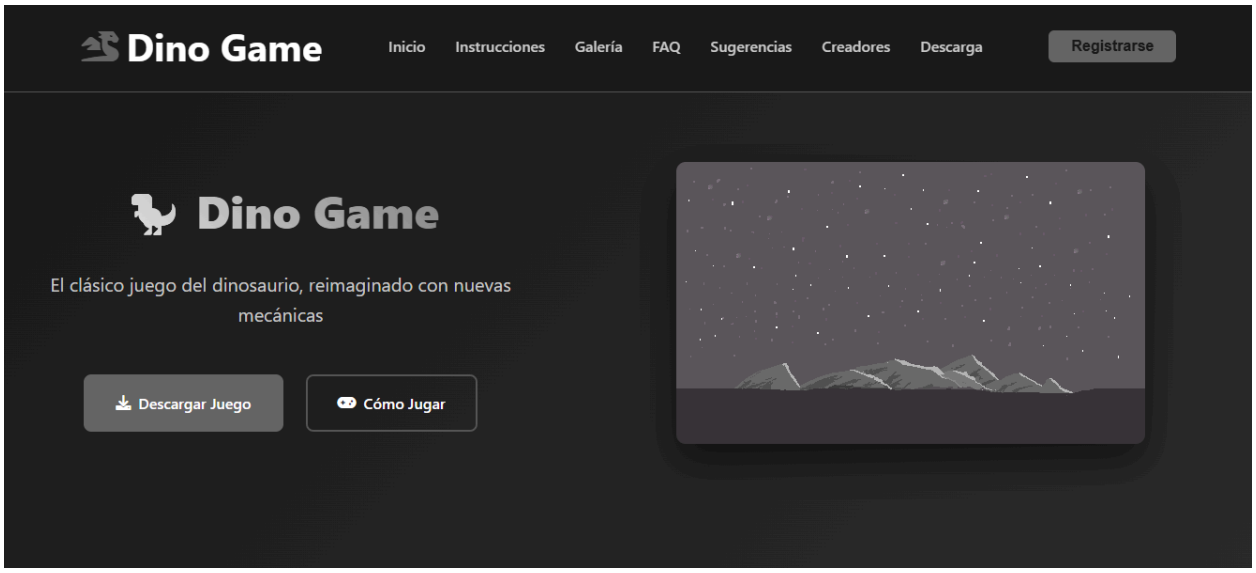
7.4.3.4 Puntaje

El jugador acumula puntos por distancia recorrida y por superar obstáculos.

El juego termina cuando el dinosaurio choca con un obstáculo. Se muestra una pantalla de fin de partida con el puntaje obtenido y se guarda el resultado en el ranking

8. Página Web

8.1. Inicio



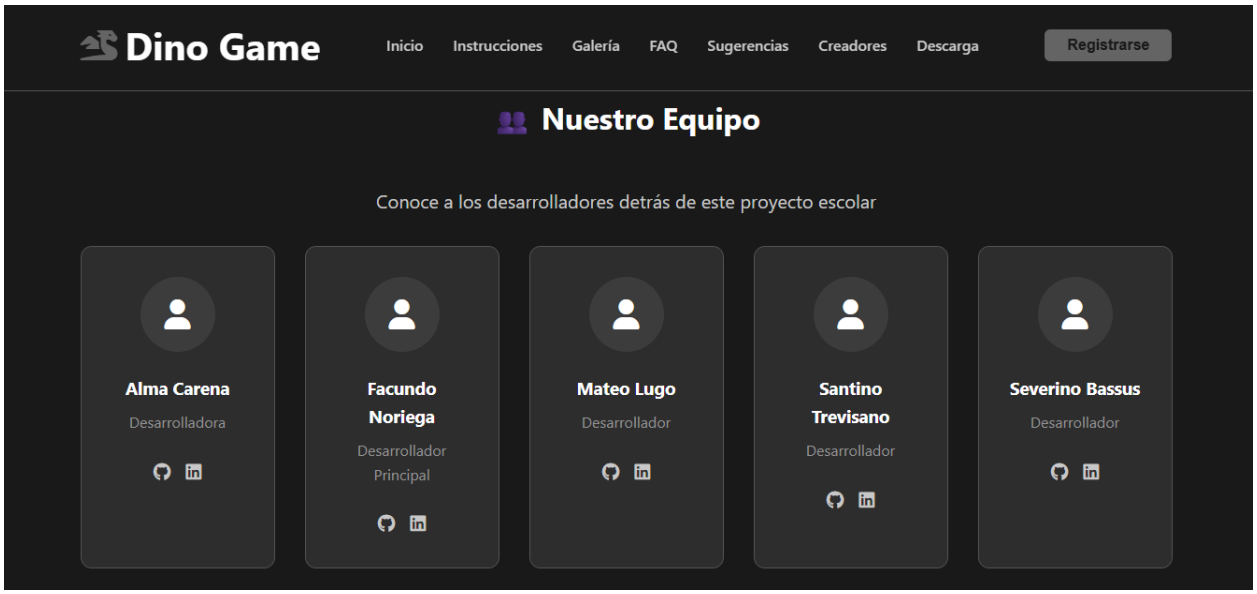
8.2. Instrucciones

8.3. Galería imágenes

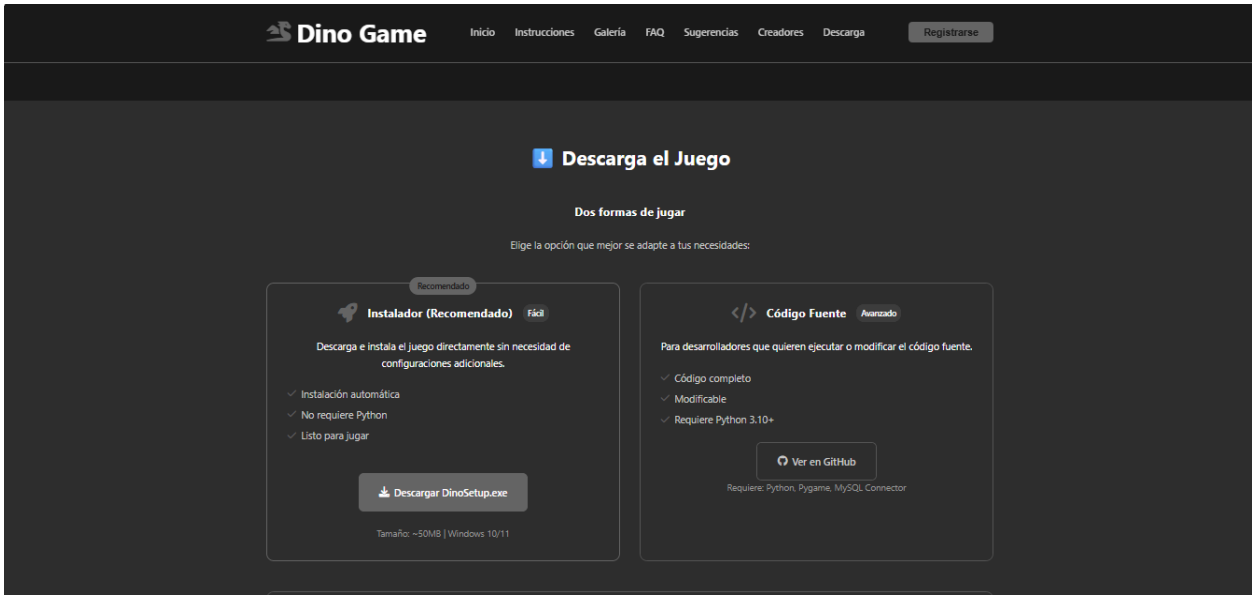
8.4. Preguntas frecuentes (FAQ)

8.5. Sugerencias y feedback

8.6. Creadores



8.7. Descarga


Dino Game
Inicio
Instrucciones
Galería
FAQ
Sugerencias
Creadores
Descarga
Registrarse
Descarga el Juego
Dos formas de jugar
Elige la opción que mejor se adapte a tus necesidades:
Recomendado
Instalador (Recomendado)
Fácil
Descarga e instala el juego directamente sin necesidad de configuraciones adicionales.
Instalación automática
No requiere Python
Listo para jugar
Descargar DinoSetup.exe
Tamaño: ~50MB | Windows 10/11
Código Fuente
Avanzado
Para desarrolladores que quieren ejecutar o modificar el código fuente.
Código completo
Modificable
Requiere Python 3.10+
Ver en GitHub
Requiere: Python, Pygame, MySQL Connector

9. Manual del usuario

Esta sección corresponde al Manual del Usuario, el cual tiene como objetivo explicar de manera clara y sencilla el funcionamiento del mando de juego y su interacción con el sistema. Aquí se detallan los pasos que debe seguir el usuario para utilizar correctamente el dispositivo, las funciones de cada botón y las posibles situaciones que pueden presentarse durante su uso.

También se incluyen las recomendaciones básicas de uso, con el fin de garantizar una experiencia de juego estable, segura y sin errores.

9.1 Manual de usuario en Español.

Consulta en el siguiente enlace para ver el **ManualDelUsuario**

9.2 Manual de usuario en Inglés.

Consulta en el siguiente enlace para ver el **UserManual**

Referencias

Pixilart. Plataforma en línea utilizada para la creación de sprites, fondos y animaciones del videojuego. Disponible en: https://www.pixilart.com

Visual Studio Code (VS Code). Entorno de desarrollo utilizado para programar el videojuego y los módulos de conexión con Arduino y la base de datos. Disponible en: https://code.visualstudio.com

ChatGPT (OpenAI). Herramienta de inteligencia artificial utilizada para el desarrollo del código base, redacción del informe y asistencia técnica. Disponible en: https://chat.openai.com

Gemini (Google AI). Inteligencia artificial empleada para la generación y retoque de imágenes del juego y los personajes. Disponible en: https://gemini.google.com

Arduino IDE. Entorno de programación utilizado para cargar y depurar el código del microcontrolador Arduino Nano. Disponible en: https://www.arduino.cc/en/software

Tinkercad. Plataforma en línea utilizada para simular y probar el circuito electrónico del mando de juego. Disponible en: https://www.tinkercad.com

Master PLC. Software empleado para representar y diseñar el circuito lógico del sistema de control táctil.

MySQL Workbench. Herramienta utilizada para diseñar el modelo entidad-relación y gestionar la base de datos del ranking de jugadores. Disponible en: https://www.mysql.com/products/workbench/

Python 3. Lenguaje de programación utilizado para desarrollar el videojuego, integrar el control Arduino y gestionar la base de datos. Documentación oficial: https://www.python.org/doc/