



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA EN SOFTWARE

Periodo Académico: 2025-B

Fecha de Entrega: 31/01/2026

Asignatura: Desarrollo de Juegos Interactivos

Grupo: GR1SW

Integrantes: Guerra Sebastián, Morales Anthony, Proaño Estéfano

Título Provisional: Protocol: SILENCE

Género: Sci-Fi Survival Horror / Sigilo Técnico

Fórmula de Identidad (X meets Y):

"Alien: Isolation meets Wall-E (con mecánicas de gestión de recursos)."

Protocolo: SILENCIO es un juego de supervivencia y sigilo isométrico donde el jugador controla a una unidad de mantenimiento (Dron Modelo M-NT) con batería crítica. El objetivo es reactivar los servidores de soporte vital de una nave colonial a oscuras, evitando ser detectado por los sistemas de seguridad corruptos que reaccionan a fuentes de sonido y luz.

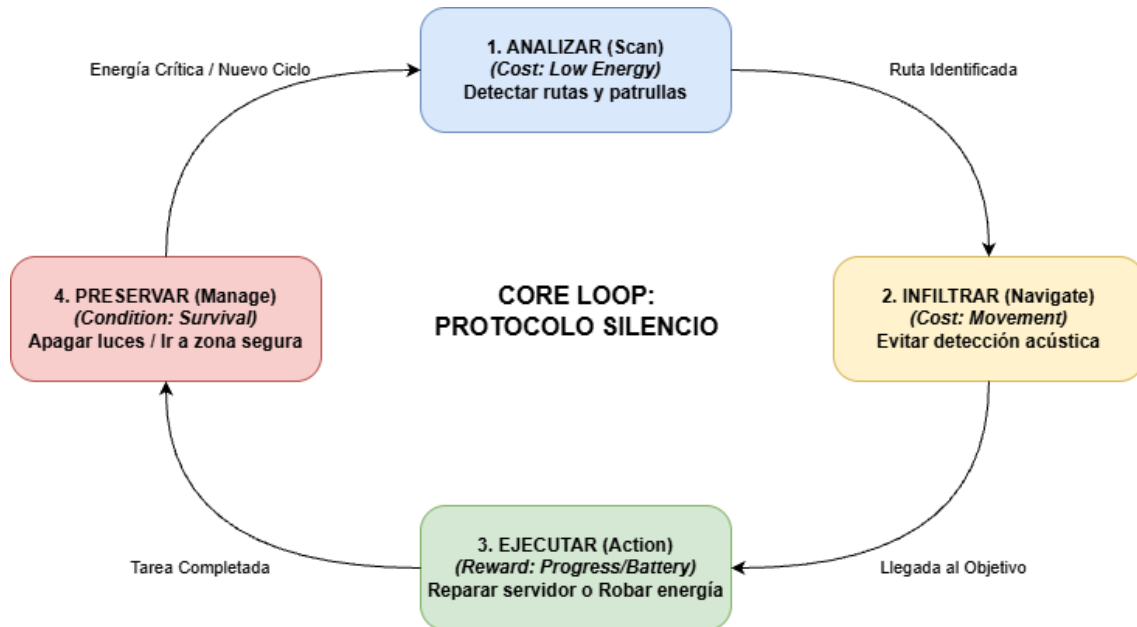
USPs (Unique Selling Points): Estas son las características sistémicas que diferenciaremos en el código:

1. **Sistema de Energía Unificado (The Life-Battery):** No existe barra de salud separada. La batería alimenta el movimiento, la linterna y la integridad del chasis.
 - *Impacto Técnico:* Usar la linterna para ver mejora la visibilidad pero reduce el "tiempo de vida" restante (Risk/Reward System).
2. **Sigilo Acústico Determinista:** Los enemigos no patrullan aleatoriamente; son sensores de audio con patas.
 - *Impacto Técnico:* El radio de detección se calcula en tiempo real basado en:
Velocidad del Jugador * Multiplicador de Superficie (Metal vs Goma).
3. **Interfaz 100% Diegética:** No hay HUD superpuesto.

- *Impacto Técnico:* La batería se muestra en la espalda del dron (luces LED) y los avisos de peligro son proyecciones holográficas del propio personaje en el mundo 3D.

CORE Loop

Este es el diagrama de flujo de la experiencia segundo a segundo.



MDA

A continuación, se definen las reglas exactas para que los programadores no tengan dudas. Usaremos el formato de tabla recomendado en la guía.

Mecánica A: Locomoción y Emisión Acústica

Estado	Velocidad (u/s)	Radio de Ruido (m)	Consumo Energía (E/s)	Visibilidad
Idle (Reposo)	0	0.0 m	0.5 (Sistemas base)	Baja
Hover (Sigilo)	3.5	2.5 m	1	Media
Turbo (Sprint)	9	18.0 m	4	Alta (Genera calor/luz)
Overload (Daño)	1	12.0 m (Chispas)	5.0 (Fuga)	Muy Alta

Regla de Superficie: El suelo de "Rejilla Metálica" aplica un multiplicador de 1.5x al Radio de Ruido. El suelo de "Goma Aislante" aplica

Mecánica B: Sistema de Visión y Luz

- **Estado Base (Apagado):** El jugador solo ve un radio de 2 metros alrededor del dron (Fog of War densa).
- **Linterna Activa (Input: F):**
 - Proyecta un cono de luz de 45 grados y 10 metros de alcance.
 - **Costo:** +1.5 E/s adicionales.
 - **Riesgo:** Los enemigos detectan la luz instantáneamente si impacta sus sensores ópticos.

Mundo Y Diseño De Niveles

1. Narrativa Ambiental (Environmental Storytelling)

Objetivo: Comunicar el estado crítico de la nave y la amenaza sin usar cuadros de texto.

- **El "Asset" Narrativo Principal: Cables de Datos Cortados.**
 - *Ubicación:* Pasillos principales y base de los servidores.
 - *Mensaje Implícito:* El daño no es por deterioro natural, es sabotaje o ataque. Algo *mordió* o *arrancó* la infraestructura.
- **Iluminación Diegética:**
 - **Luz Azul (Estática):** Zonas seguras (Sistemas funcionando).
 - **Luz Roja (Pulsante):** Zonas de patrulla enemiga (Alarma de seguridad).
 - **Oscuridad Total:** Zonas de riesgo desconocido (Requiere linterna -> Costo de Batería).
- **Marcas de Arrastre:** Texturas de "arañazos" en el suelo metálico que conducen hacia las rejillas de ventilación (sugiere que los técnicos anteriores fueron arrastrados hacia allí).

2. Diseño de Nivel 1: Plano Top-Down ("Cooling Unit 04")

He diseñado la arquitectura para forzar las mecánicas de **Sigilo** y **Gestión de Ruido**.

Descripción del Layout para tu dibujo: El nivel tiene forma de "U" invertida, obligando al jugador a rodear la zona central peligrosa.

1. **Zona de Inicio (Safe Zone):** *El Conducto de Ventilación Norte.* Espacio estrecho, baja altura. El jugador está seguro aquí para probar los controles (Moverse, Linterna).

2. **El Umbral (Threshold):** *Puerta de Mantenimiento A-01*. Está atascada a media altura. El jugador debe agacharse (Crouch) para pasar, entrando oficialmente al área de juego.
3. **Zona Principal (The Killbox):** *Pasillo Central de Servidores*.
- **Suelo:** Rejilla metálica (Ruidoso Multiplicador x1.5).
 - **Enemigo:** Un "Sensor Bot" patrulla en bucle infinito de un extremo al otro.
 - **Coberturas:** Racks de servidores (Cajas altas) que bloquean la línea de visión pero no el sonido.
4. **Ruta Alternativa (Stealth Path):** Un camino lateral estrecho con suelo de goma (Silencioso), pero bloqueado por vapor (daño leve si no se espera el timing correcto).
5. **El Objetivo (Goal):** *Terminal Principal*. Ubicada al fondo de la "U". Requiere 5 segundos de interacción continua.

3. Tabla Técnica de Eventos (Level Scripting)

Esta tabla define la lógica que los programadores implementarán en el motor (Unity/Unreal/Babylon).

ID Evento	Trigger (Disparador)	Condición Lógica	Acción (Output)
EV_01_Intro	OnTriggerEnter(Zone_Start)	GameStart == true	Audio.Play("SystemBoot"); Light.Blink(Red, 3s);
EV_02_Noise	OnStep(Surface_Metal)	Player.Speed > 3.0 (Run)	EnemyAI.SetState(ALERT); Audio.Play("MetalClang_Loud");
EV_03_Steam	OnTriggerStay(Zone_Steam)	Timer % 3 == 0 (Ciclo vapor)	Player.Battery -= 5.0; VFX.Play("SteamBurn");
EV_04_Hack	OnInteract(MainTerminal)	Input.HoldTime >= 5.0s	Level.ObjectiveComplete = true; Lights.SetAll(Blue);

EV_05_Escape	OnTriggerEnter(Vent_Exit)	ObjectiveComplete == true	GameManager.LoadLevel(2);
--------------	---------------------------	---------------------------	---------------------------

4. Plan de Greyboxing (Prototipado)

Para probar esto antes de hacer el arte final, necesitamos definir las primitivas geométricas.

Leyenda de Colores para el Greybox:

- **Cubos Grises (1x1m):** Suelo estándar (Ruido normal).
- **Cubos Rojos:** Suelo de Metal (Ruido Alto - Zona de peligro).
- **Cubos Azules:** Suelo de Goma (Ruido Nulo - Ruta segura).
- **Cilindro Verde:** El Jugador (Hitbox del dron).
- **Esfera Naranja:** El Enemigo (Sensor).
- **Bloques Negros (2x1m):** Coberturas (Servidores) - *Collision Only, No Rendering.*

Tarea Inmediata de Diseño: Construir este circuito usando solo estas formas simples para validar si el tiempo que tarda el enemigo en patrullar permite al jugador cruzar por la ruta de goma.

5. Product Management, UX y Balanceo del Sistema

5.1. Estética y Dirección de Arte

Paleta de colores

En Protocol: SILENCE, la luz es información. No usamos colores por estética, sino por función (Game Design).

- **System_OK (Azul Glacial): #00F0FF**
 - Uso: Zonas seguras, batería > 50%, terminales hackeados.
 - Emoción: Frialdad tecnológica, seguridad, calma.



- **System_Warning (Ámbar Industrial): #FFAE00**
 - Uso: Batería < 50%, objetos interactivos, zonas de ruido moderado (metal).
 - Emoción: Precaución, deterioro, suciedad.



- **System_Critical (Rojo Alarma): #FF2A00**
 - Uso: Enemigos (ojos), Batería < 15%, Daño recibido.
 - Emoción: Pánico, urgencia, error fatal.



- **The Void (Negro Profundo): #050505**
 - Uso: Niebla de guerra, zonas sin linterna.
 - Emoción: Lo desconocido, claustrofobia.



Dirección de Audio (Soundscape)

- **El Silencio (Base):** No es silencio total. Es un "Hum" grave de ventiladores lejanos (Low frequency rumble) + crujidos metálicos aleatorios.
- **El Jugador (Diegetic):** Zumbido de servos al moverse. Al correr, sonido de pistones hidráulicos chocando.

- **El Enemigo:** Sonido de "clic-clic-clic" (como un disco duro roto) que aumenta de frecuencia según la proximidad.

5.2. Interfaz de Usuario

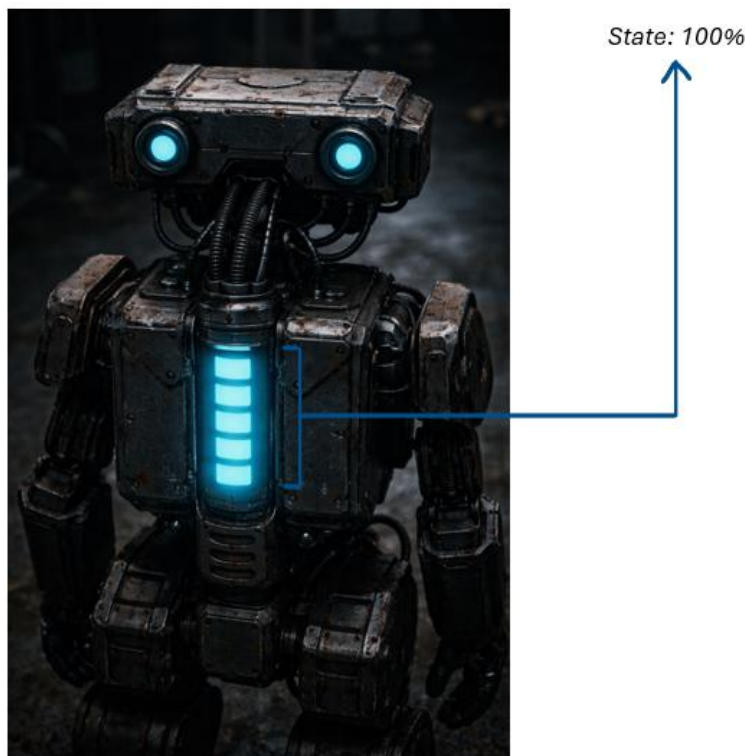
El sistema de interfaz de usuario de Protocol: SILENCE elimina por completo el HUD (Heads-Up Display) tradicional superpuesto en pantalla. Siguiendo los principios de inmersión, toda la información vital se comunica a través de elementos visuales integrados en el modelo 3D del personaje y el entorno.

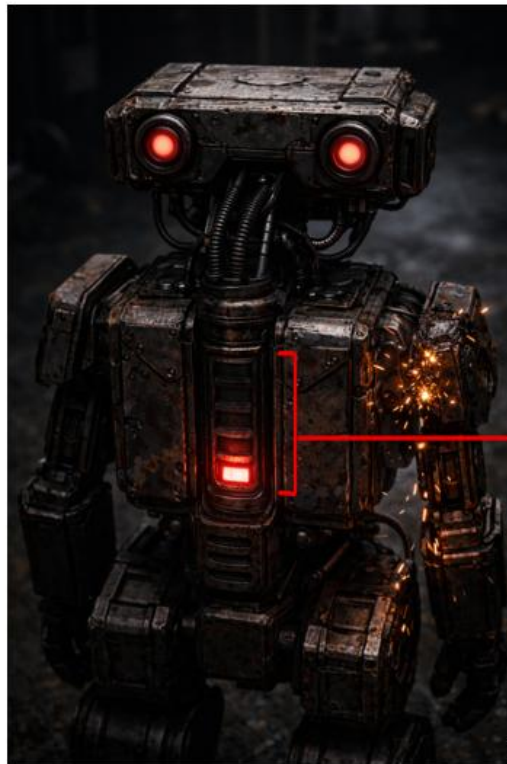
5.2.1. Sistema de Visualización de Estado (The Life-Battery)

La gestión de energía es el núcleo del gameplay. Para comunicar este valor sin barras de vida flotantes, utilizamos un array de emisores de luz (Emissive Materials) en la columna dorsal del dron M-NT.

Estados Visuales Definidos:

- **Estado Óptimo (100% - 50%):** El indicador muestra 5 segmentos iluminados en Azul Glacial (#00F0FF). Indica que el jugador puede usar la linterna y el sprint sin riesgo inmediato.
- **Estado Crítico (< 20%):** El indicador se reduce a 1 segmento en Rojo Alarma (#FF2A00) con un efecto de parpadeo (Strobing). Además, se activan sistemas de partículas de chispas (Sparks VFX) en el hombro derecho para indicar daño estructural severo.





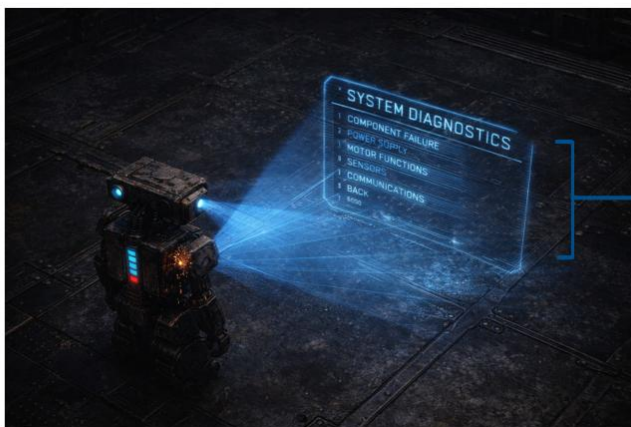
State: Critical < 20%
Particle System: Sparks (Loop)

Mockup de alta fidelidad mostrando la transición de estado. A la izquierda, unidad operativa

5.2.2. Interfaz de Gestión Holográfica (Inventario)

Para evitar romper la tensión del juego, el menú de inventario no pausa la acción.

- **Mecánica de Proyección:** El dron proyecta un cono de luz volumétrica hacia el suelo (Raycast desde los ojos hacia la normal del terreno).
- **Vulnerabilidad (Risk/Reward):** Al mirar el inventario, el jugador debe detenerse y mirar hacia abajo, perdiendo visión periférica de los enemigos. La luz azul de la proyección también aumenta ligeramente el radio de detección visual, haciendo que "gestionar el inventario" sea una decisión táctica peligrosa.



World Space Canvas (No Pausa)

Prototipo de visualización del menú "System Diagnostics". La interfaz se renderiza en el espacio del mundo (World Space Canvas)

5.3. Balanceo y Economía del Juego

El núcleo de la experiencia de Protocol: SILENCE reside en la escasez. Para garantizar que el juego genere tensión sin llegar a ser frustrante, se ha diseñado un modelo matemático de Fuentes y Drenajes (Sources & Sinks) que regula el flujo de energía.

A continuación, se documentan las variables constantes que alimentarán el archivo de configuración `battery_config.json` y la simulación de viabilidad del Nivel 1.

5.3.1. Definición de Constantes del Sistema

Se han definido dos tablas maestras para regir el comportamiento de la batería. Estas variables están separadas del código lógico para permitir ajustes rápidos de balanceo (tweaking) sin recompilar el proyecto.

- **Tabla de Drenajes (Sinks):** Define el costo por segundo de cada acción. Nótese el multiplicador de terreno para superficies difíciles.
- **Tabla de Fuentes (Sources):** Define cómo el jugador recupera "vida", estableciendo el ritmo de recompensa.

Tabla A: Constantes de Consumo (Sinks)

Variable ID	Valor Base	Unidad	Notas
BASE_IDLE	0.2	E/s	Consumo mínimo.
MOV_WALK	1.0	E/s	Standard movement.
MOV_SPRINT	4.0	E/s	Emergencia.
MOD_FLASH LIGHT	+1.5	E/s	Se suma al movimiento.
MOD_TERRAIN_MUD	x 1.2	Multiplier	Suelo difícil gasta más batería.

Tabla B: Constantes de Recuperación (Sources)

Variable ID	Valor (Gain)	Cooldown	Mecánica
ITEM_BATTERY_S	+15.0 E	Instant	Batería pequeña (consumible).
ITEM_BATTERY_L	+40.0 E	Instant	Batería grande (rara).
STATION_CHARGE	+10.0 E/s	N/A	Estación de pared (requiere estar quieto).

Constantes globales de consumo y recuperación. Valores sujetos a cambios tras el Playtesting de la versión Alpha.

5.3.2. Simulación de Escenario: "Pasillo de Servidores"

Para validar matemáticamente el diseño de nivel antes de la implementación, se realizó una simulación de estrés ("Stress Test") de 20 segundos.

Condiciones del Test:

- Escenario: El jugador debe cruzar una zona de patrulla enemiga.
- Estado Inicial: Batería al 50%.
- Objetivo: Demostrar que el uso descuidado de recursos (Sprint + Linterna) lleva al jugador al estado crítico, forzando una decisión táctica.

Escenario: Cruzando el "Pasillo de Servidores" (Nivel 1)

Condiciones Iniciales: Batería al 50% (50 Unidades).

T (seg.)	Acción Jugador	Estado Linterna	Consumo (Calc)	Evento Externo	Batería Final	Límite Crítico	Análisis
0	Inicio	OFF	0	-	50,00	20	Inicio seguro.
5	Caminar (Walk)	OFF	5	-	45,00	20	Desplazamiento a oscuras.
10	Caminar	ON	12,5	-	32,50	20	Enciende luz para ver trampas.
12	SPRINT	ON	11	Enemigo Detecta!	21,50	20	Pánico. Gasto masivo en 2 seg.
15	Crouch (Esconder)	OFF	1,5	-	20,00	20	Entra en estado Crítico (Rojo).
20	Idle (Quieto)	OFF	1	Loot (+15.0)	34,00	20	Encuentra pila pequeña. Salavado.

Tabla de datos de la simulación paso a paso. Se observa el impacto masivo del evento "Enemigo Detecta" en el segundo 12.

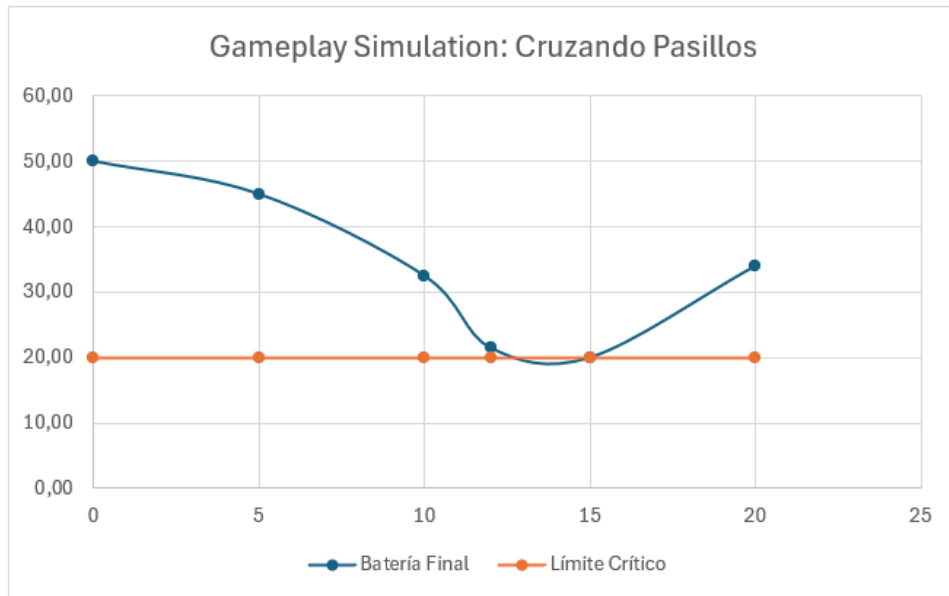
5.3.3. Análisis de la Curva de Tensión

La visualización de los datos de la simulación revela el comportamiento deseado de la dinámica de juego.

Interpretación de la Curva:

La línea azul (Energía Actual) muestra una caída pronunciada entre T=10s y T=12s. Esto confirma la hipótesis de diseño: el jugador no puede mantener la linterna y el sprint activos simultáneamente por más de unos segundos sin consecuencias graves.

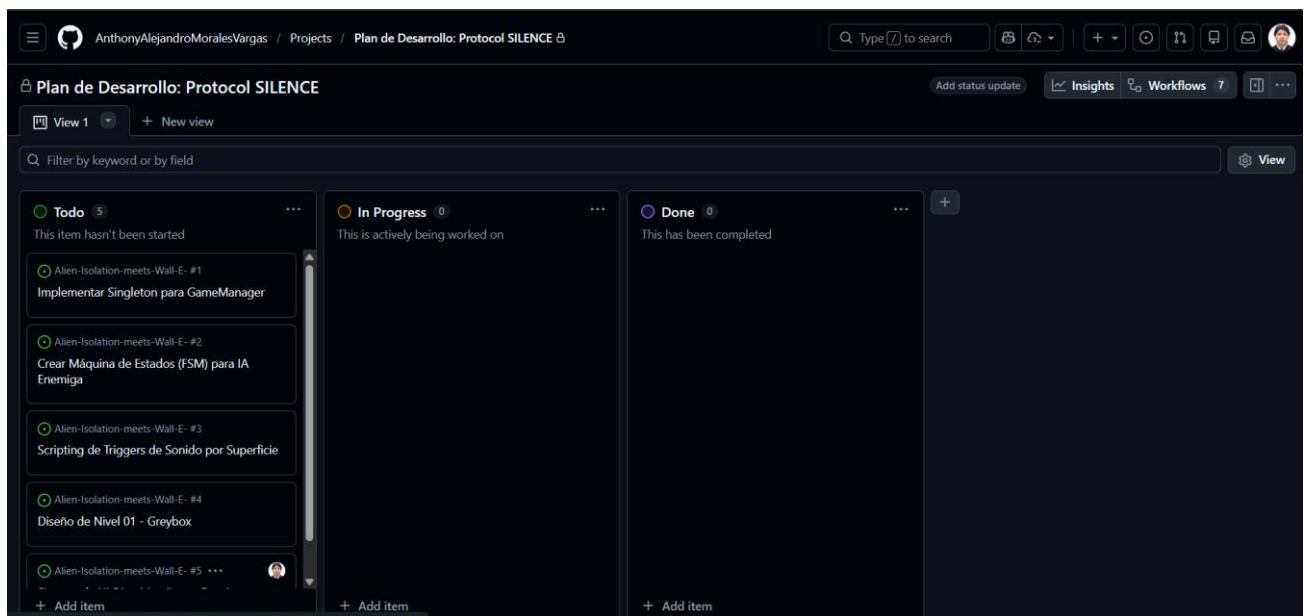
El punto de intersección en T=15s es crítico: la energía toca la línea naranja (Umbral Crítico = 20), activando los sistemas de emergencia (UI Roja/Audio de Alarma). El sistema está balanceado para que el jugador sobreviva apenas con lo justo antes de encontrar el recurso de recuperación (Loot), maximizando la respuesta emocional (Aesthetics).



Gráfica de dispersión mostrando la Curva de Supervivencia. La línea naranja representa el "Hard Limit" donde el dron entra en modo de fallo sistémico

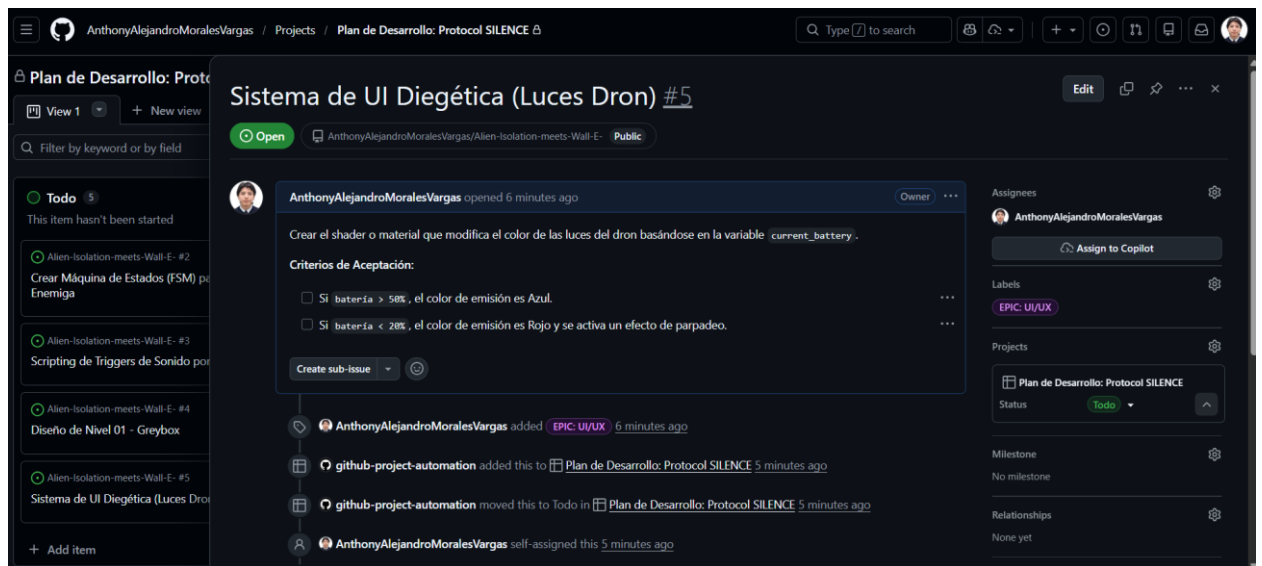
5.4. Gestión del Proyecto

Para la gestión del ciclo de vida del proyecto, se adoptó una metodología ágil basada en un tablero Kanban, implementado a través de la herramienta GitHub Projects. El trabajo se ha desglosado en Épicas y Tareas (Issues) asignadas a cada miembro del equipo.



Tablero Kanban del proyecto con el backlog inicial en la columna "To Do".

Cada tarea incluye una descripción clara, un responsable y Criterios de Aceptación para validar su finalización, asegurando que el entregable cumpla con los requisitos de diseño.



Ejemplo de una Tarea (Issue) con sus componentes: Asignado, Etiqueta de Épica y Criterios de Aceptación claros.

6. Arquitectura de Software (Ingeniería)

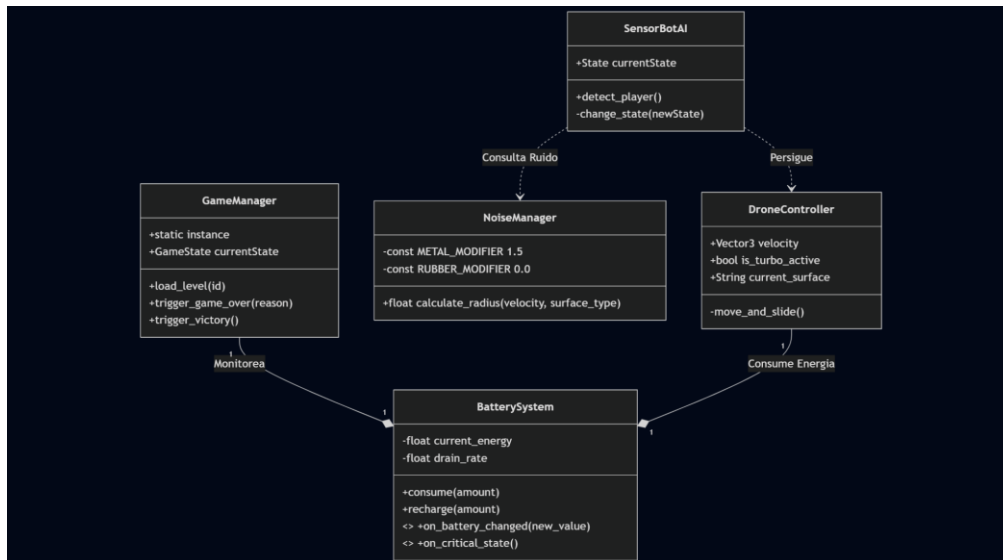
6.1. Stack Tecnológico

- **Motor:** Godot Engine 4.x. *Justificación:* Gestión nativa de unidades métricas reales para física (necesario para el cálculo de ruido en metros) y sistema de nodos ligero para la IA.
- **Control de Versiones:** Git + GitHub Desktop. Estructura de ramas: *main* (producción), *develop* (integración) y *feature/mechanic_name*.
- **IDE:** Visual Studio Code con extensión *godot-tools*.
- **Formato de Assets:** *.gltf* (Modelos 3D), *.png* (UI), *.ogg* (Audio comprimido).

6.2. Arquitectura del Núcleo (Diagrama de Clases)

El sistema utiliza una arquitectura orientada a componentes desacoplados. El *GameManager* orquesta el flujo, mientras que los sistemas críticos (*BatterySystem*, *NoiseManager*) operan de forma independiente para evitar dependencias circulares.

6.2.1. Diagrama UML de Clases



6.3. Patrones de Diseño

Se implementan estrictamente tres patrones de diseño para garantizar la escalabilidad.

1. Patrón Singleton (GameManager)

- **Justificación:** Se requiere una única "fuente de verdad" para el estado de la partida. Evita errores comunes como tener múltiples instancias cargando el nivel al mismo tiempo o solapamiento de audio.

2. Patrón Observer (BatterySystem to UI)

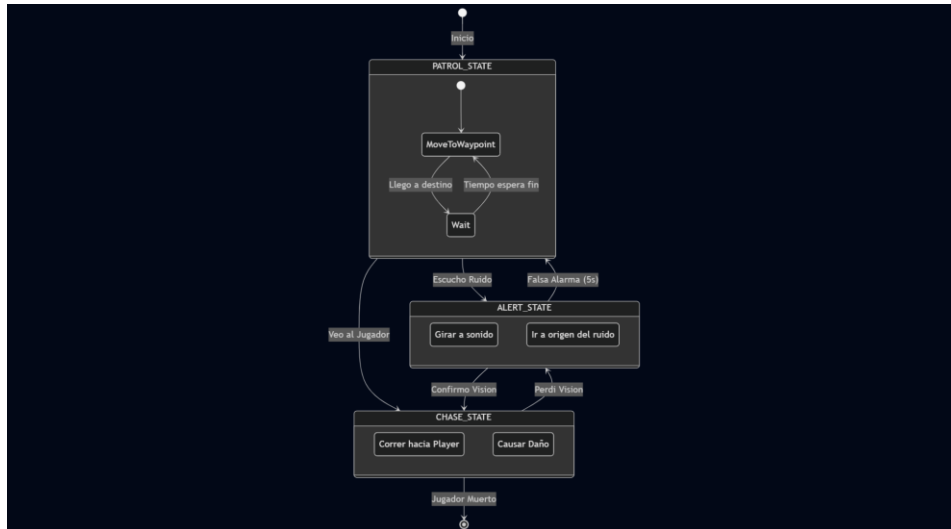
- **Problema:** La interfaz visual (luces en el dron) necesita saber la batería restante, pero la lógica no debe depender de los gráficos.
- **Solución:** El BatterySystem emite una señal `on_battery_changed(value)`. La UI se "suscribe" a esta señal y reacciona automáticamente.
- **Beneficio:** Desacoplamiento total. Si cambiamos el modelo 3D del dron, la lógica de batería no se rompe.

3. Patrón State (Máquina de Estados de IA)

- **Justificación:** El enemigo tiene comportamientos mutuamente excluyentes (Patrullar vs. Perseguir). Gestionarlos con `if/else` crea errores lógicos. Este patrón encapsula cada comportamiento en una clase propia.

7.3.2. Diagrama de Estados del Enemigo (FSM)

Lógica de transición del *Sensor Bot* basada en las mecánicas de ruido y visión.



6.4. Diseño de Sistemas y Datos

6.4.1. Backend de Lógica de Sigilo (NoiseCalculator)

Implementación de la fórmula matemática solicitada ("Peso vs Velocidad" adaptado a "Ruido vs Velocidad"). Esta clase estática determina la detección.

```

1  class_name NoiseCalculator
2
3  # Constantes de Ruido Base (Según GDD Fase 1)
4  const NOISE_IDLE = 0.0      # Quieto
5  const NOISE_HOVER = 2.5     # Velocidad 3.5 u/s
6  const NOISE_TURBO = 18.0    # Velocidad 9.0 u/s
7
8  # Multiplicadores de Superficie
9  const MOD_METAL = 1.5       # Rejilla ruidosa
10 const MOD_RUBBER = 0.0      # Goma (Silencio)
11
12 static func calculate_detection_radius(velocity_mag: float, surface_type: String) -> float:
13     var base_noise = 0.0
14
15     # 1. Determinar Ruido Base
16     if velocity_mag > 8.0:
17         base_noise = NOISE_TURBO
18     elif velocity_mag > 0.1:
19         base_noise = NOISE_HOVER
20     else:
21         base_noise = NOISE_IDLE
22
23     # 2. Aplicar Modificador de Superficie
24     var modifier = 1.0
25     if surface_type == "Metal": modifier = MOD_METAL
26     if surface_type == "Rubber": modifier = MOD_RUBBER
27
28     return base_noise * modifier
  
```

6.4.2. Persistencia de Datos (JSON Configuration)

Para permitir el balanceo sin modificar código, los valores de la "Life-Battery" se externalizan.

Archivo: *battery_config.json*

```

1 {
2   "drone_stats": {
3     "max_battery": 100.0,
4     "critical_threshold": 15.0
5   },
6   "consumption_rates_per_second": {
7     "idle": 0.5,      // Gasto base
8     "hover": 1.0,     // Gasto al moverse
9     "flashlight": 1.5, // Costo por usar luz
10    "turbo": 4.0,      // Costo por sprint
11    "damage_leak": 5.0 // Fuga por daño
12  }
13 }

```

6.5. Restricciones de Arte Técnico

- **Presupuesto de Polígonos:**
 - Player (Dron): Máx. 5,000 tris.
 - Enemigo: Máx. 3,000 tris.
- **Iluminación:** Máximo 3 luces dinámicas con sombras (*Shadow Casting*) simultáneas. Prioridad 1: Linterna del jugador.
- **Estructura de Directorios:**

